Web服务与数据交换关键技术研究

WEB FUWU YU SHUJU JIAOHUAN GUANJIAN JISHU YANJIU

马晓轩 / 著

中国环境出版社

Web服务与数据交换关键技术研究

马晓轩　著

中国环境出版社·北京

图书在版编目（CIP）数据

Web 服务与数据交换关键技术研究/马晓轩著. —北京：中国环境出版社，2017.3

ISBN 978-7-5111-2839-3

Ⅰ. ①W… Ⅱ. ①马… Ⅲ. ①Web 服务器—研究 ②数据交换—研究 Ⅳ. ①TP393.092.1②TN919.6

中国版本图书馆 CIP 数据核字（2017）第 040145 号

出 版 人 王新程
责任编辑 侯华华
责任校对 尹 芳
封面设计 宋 瑞

更多信息，请关注
中国环境出版社
第一分社

出版发行 中国环境出版社
（100062 北京市东城区广渠门内大街 16 号）
网 址：http://www.cesp.com.cn
电子邮箱：bjgl@cesp.com.cn
联系电话：010-67112765（编辑管理部）
联系电话：010-67112735（第一分社）
发行热线：010-67125803，010-67113405（传真）
印 刷 北京中献拓方科技发展有限公司
经 销 各地新华书店
版 次 2017 年 3 月第 1 版
印 次 2017 年 3 月第 1 次印刷
开 本 880×1230 1/32
印 张 5
字 数 115 千字
定 价 26.00 元

前 言

传统客户/服务器的分布式计算模式主要解决的是部门或企业内部的应用，关注的是局部应用，建立的应用系统多数是为完成某种需求而建立，因此是孤立封闭的。即使在同一个部门或者企业内部，应用系统相互之间也常常因为没有通信而导致信息无法共享，使得各个 IT 资源成为一个个信息孤岛。随着互联网软件技术及其应用的迅速发展，如何提供一个统一开放的交互环境，使得各个应用实体之间能够相互发现、了解各自所提供的服务，并将这些应用低代价、方便地连接在一起，实现开放式网络环境中的互联、互通、互操作的目标，是当前互联网环境下所面临的一个重要问题。

想要有效地整合现有的应用系统并实现信息共享，关键在于实现这些应用系统间的数据交换。数据交换是网络环境下分布式应用的共性、基础性和关键性的需求，它用于解决信息化过程中数据资源的互操作问题，即解决不同的异构系统之间的

数据资源的整合和共享。Web 服务的目的和作用是提供一种国际统一的规范和技术，进行 Internet 上各种软件应用的统一功能描述和功能共享，为功能整合集成和信息交换处理提供实现基础。本书分析了大规模数据交换在 Web 应用中所面临的问题，针对数据交换平台框架、数据交换的可靠性、数据交换的效率以及数据服务中心等一系列关键技术展开研究，主要的研究结果如下：

（1）设计了基于 Web 服务的数据交换框架。分析了使用 Web 服务进行数据交换的必要性以及数据交换的应用模式。提出基于 Web 服务的数据交换平台框架，并由此提出基于 Web 服务的大规模数据交换的若干关键技术。

（2）提出了基于 Web 服务的数据交换可靠消息机制。通过分析数据交换的可靠性因素，从数据交换消息层定义了消息可靠性，讨论了消息可靠性的参数，提出一种可靠的消息传输机制，主要从消息确认、消息重复、消息丢失三个方面来确保数据交换的可靠性，并从可靠消息处理机和可靠报文结构两个方面阐述了该机制的实现。

（3）提出了两种改进基于 Web 服务的数据交换传输效率的机制。首先分析了影响 Web 服务性能的因素，提出了 Web 服务性能的优化方案，在此基础上提出了基于 Web 服务的类

型映射及基于滑动窗口的 SOAP(Simple object access protocol，简单对象访问协议）消息传输等两种改进数据交换的传输效率的机制。针对 Web 服务的性能优化需求，基于 WSDL（Web services description language，Web 服务描述语言）文档校验机制，提出了一种可扩展的高效类型映射机制；基于滑动窗口的 SOAP 消息传输机制通过将数据切片，实现对大数据和可变长度数据的传输优化，并通过设计滑动窗口协议来进一步提高 SOAP 消息的传输速度；提出了基于应用层的拥塞控制算法，并对影响传输效率的两个主要因素（滑动窗口大小和数据切片长度）进行了测试。

（4）设计了基于分布式信息存储模型的数据服务中心。在对数据交换中数据服务的可扩展性、发现效率和有效性进行分析的基础上，设计了一个分布式信息存储模型，讨论了注册中心的组织，提出了信息动态注册与更新机制，设计了一个分布式信息搜索算法，并设计实现了基于 UDDI（Universal description，discovery，and integration，通用描述发现和集成）的数据服务注册中心 DSC。针对数据服务中心的安全及管理等问题，对原有的 UDDI 协议进行了一些扩展，提出了发布者声明关联匹配算法，发布者声明关联匹配算法能够描述注册应用之间的关联关系。

（5）设计并实现了基于Web服务的数据交换平台。提出了一个数据交换协议，在此基础上，设计实现了基于Web服务的数据交换平台，讨论了其系统的总体结构，阐述了其中的关键机制如事务、适配器等，并且讨论了数据交换平台在电子政务中的应用部署。

目 录

第 1 章
概　论

1.1　异构系统数据交换面临的挑战

随着计算技术，尤其是近年来网络技术和应用的不断发展，计算模式也在不断发生变化，从早期的单一主机、主机/终端模式演化到以构件技术和分布式对象技术为代表的客户/服务器模式。与此同时，大量的业务应用也开始向互联网迁移。由于传统客户/服务器的分布式计算模式主要解决的是部门或企业内部的应用，关注的是局部应用，建立的应用系统多数是为完成某种需求而建立，因此是孤立的、封闭的，即使在同一个部门或者企业内部，应用系统相互之间也常常因为没有通信导致信息无法共享，使得各个 IT 资源成为一个个信息孤岛，相互间壁垒森严，最终效率低下。随着社会信息化程度的不断提高，信息孤岛的制约作用越来越突出，导致信息技术不能更好地满足社会发展的需求。随着市场经济的发展，政府部门面临转变政府职能、提高行政能力、服务社会公众的挑战；而企业用户也随着业务规模的扩大和产品质量的不断提升，迫切需要提高业务运作的效率、快速用户响应，以提高用户的满意度。为适应这

些新形势的发展，迫切需要打破现有应用系统之间的信息隔离，整合和共享已有的信息资源，以提高应用系统间的信息共享和协同工作。如何提供一个统一开放的交互环境，使各个应用实体之间能够相互发现、了解各自所提供的服务，并将这些应用低代价、方便地连接在一起，实现开放式网络环境中的互联、互通、互操作的目标，是当前互联网环境下所面临的一大问题。

为有效地整合现有的应用系统并进行信息共享，关键在于解决这些应用系统间的数据交换。数据交换是网络环境下的分布式应用的共性、基础性和关键性的需求。数据交换用于解决信息化过程中数据资源的问题，即解决不同的异构系统之间的数据资源的整合和共享。

在大规模数据交换的过程中，数据交换双方可能处于同一个局域网内，也可能处于广域网上相距甚远的两个网络节点上，交换过程可能跨越多个网络自治域，不同自治域在网络状况、安全保护措施等方面都可能存在不同程度的差异；数据交换双方要交换的数据可能是文本类型的数据，如文本文件、XML（eXtensible Markup Language，可扩展标记语言）文件，也可能是图片、音频、视频、Word 文档等其他类型的文件，交换的数据可能小到只有几个字节，也可能是大到数十兆甚至上百兆数据量的多媒体数据文件。因此，由于计算机网络技术和应用的多样性，异构问题已成为当前应用系统集成的主要障碍，如何去解决异构系统间的集成以实现系统间的数据交换是资源整合的一个关键问题。

Web 服务的目的和作用是提供一种国际统一的规范和技术，进行 Internet 上各种软件应用的统一功能描述和功能共享，为功能整合集成和信息交换处理提供实现基础。通过使用 Web 服务技术，所有

的应用数据的封装都是标准化的，都是基于 SOAP 报文格式进行封装，并且使用 SOAP 消息进行传输；每个应用程序的接口描述都是使用 WSDL 来进行描述，公用数据信息使用通用描述发现集成协议来进行存储，从而使应用之间的维护、安全控制、信息过滤都变得非常容易。它不需要对遗留系统进行大的修改或重新开发。由于 Web 服务技术的出现，计算机系统之间实现统一的信息交换成为可能。

因此，基于 Web 服务技术，分析大规模数据交换所面临的问题，研究数据交换平台框架，并解决数据交换的可靠性、数据交换的效率以及数据的存储及共享等一系列关键技术，将具有重要的学术意义和应用价值。

1.2　相关技术

1.2.1　数据交换技术的发展过程

数据交换技术到目前为止大致经历了三个发展阶段：

- 第一个阶段，以电子数据交换（Electronic Data Interchange，EDI）技术为基础的数据交换。

EDI 是计算机、通信和现代管理技术相结合的产物。国际标准化组织（International Organization for Standardization，ISO）将 EDI 描述成“将贸易（商业）或行政事务处理按照一个公认的标准变成结构化的事务处理或信息数据格式，从计算机到计算机的电子传输”。而 ITU-T 将 EDI 定义为“从计算机到计算机之间的结构化的事务数据互换”。由于可以减少交易过程中的纸张使用量，EDI 被人们通俗地称为“无纸贸易”。

从EDI定义可以得出，EDI包含了三个方面的内容，即计算机应用、通信网络和数据标准化。其中，标准化的工作是实现EDI互通和互联的前提和基础。目前，在EDI标准上，国际上最有名的是联合国欧洲经济委员会（UN/ECE）下属第四工作组（WP4）于1986年制定的《用于行政管理、商业和运输的电子数据互换》标准——EDIFACT（Electronic Data Interchange For Administration，Commerce and Transport）标准。

EDIFACT标准已被国际标准化组织（ISO）接收为国际标准。同时还有广泛应用于北美地区的，由美国国家标准化协会（ANSI）X.12鉴定委员会（AXCS.12）于1985年制定的ANSIX.12标准。从目前的情况来看，EDIFACT成为统一的EDI国际标准已是大势所趋。

- 第二个阶段，以中间件技术为基础的数据交换。

中间件是基础软件的一大类，属于可复用软件的范畴。中间件运行在操作系统、网络和数据库之上，应用软件之下，为处于自己上层的应用软件提供运行与开发的环境，帮助用户灵活、高效地开发和集成日趋复杂的应用软件。中间件产品根据其作用可以分为远程过程调用、面向消息的中间件、对象请求代理和事务处理监控等。

目前用于开发中间件的主要技术有OMG的CORBA（Common Object Request Broker Architecture，公共对象请求代理体系结构），Sun的J2EE和Microsoft的DCOM（Distributed Component Object Model，分布式组件对象模型）。

在中间件的使用过程中，数据与传输紧密地耦合在一起，不便于数据的重用。同时，由于不同的中间件产品采用不同的规范，使得不同中间件之间的互操作就成了一个新的集成难题。要想解决这个问题，就必须采用统一的开放标准，使各个中间件产品依照这个

标准提供标准的接口并依据标准对接口进行描述，以方便不同中间件产品之间的相互调用。基于 XML 和 Web Services 数据交换技术可以解决这些问题。

- 第三个阶段，基于 XML 和 Web Services 技术的数据交换。

XML，Web Services 技术的出现，为人们寻找一种廉价、简单、有效的信息交换方式提供了技术基础。其中 XML 数据规范为运行于企业内部网络中不同节点的应用系统间进行数据交换奠定了基础；Web Services 又使这些应用系统相互连接并进行功能调用成为可能。

Web Services 采用 SOAP 协议可以基于 HTTP（Hyper Text Transfer Protocol，超文本传输协议）协议通信，由于 HTTP 在网络中广泛应用，为 Web Services 提供无限的互联成为可能，从而可以轻松地穿越绝大多数的防火墙，而与 XML 结合使其具有了数据交换的能力。采用基于 XML，Web Services 技术实现跨越网络的异构数据交换，也就成了理想的数据交换方式，使建设跨网络的企业应用集成成为可能。为了方便 Web 服务被描述和发现，还产生了 WSDL 和 UDDI，通过服务发现机制发现了一个 Web 服务之后就可以得到它的服务描述，有了这个描述就可以将这个功能集成到自己的应用系统中，而不用关心 Web 服务是如何构建的或是运行在什么平台之上。基于 Web Services 技术和 XML 数据结构的数据交换方法是当前数据交换平台最主要的研究方向。

1.2.2　Web Services 技术

消息服务为电子政务应用系统实现各种格式的信息从一个地方传递到另一个地方的功能服务。消息服务是一个软件，它以一种可靠的、异步的、松散耦合的、与语言无关的、与平台无关的方式在

分布式应用系统之间传递消息提供支持。

消息服务主要解决下面几个问题：

①信息的统一封装：只有统一了封装格式，才可能进行统一的交换。就像邮政信函业务，其前提条件是大家必须按规定格式填写信封。

②统一编址问题：统一的地址编码是屏蔽物理网络的内在需求，也是进行统一信息交换的基础。如何设计一套统一的、简单易用、易扩展、易管理的地址编码体系，是实现信息交换的关键。

③信息的可靠传输：消息服务是为上层应用提供信息交换服务的，所以将信息可靠地传输到目的地址是基本要求。也就是说，消息服务必须实现消息中间件的功能，而现有的IP/TCP/HTTP协议并没有实现此功能。

④路由问题：在一个大的信息交换网络中，可能会有许多信息交换节点，当某个信息需要通过多个节点到达目的地时，就出现了路由寻址的问题。

⑤传输的效率问题：信息交换关心的是信息的传输与交换，而不是信息本身的含义。为了实现高速的信息交换，需要将信息的表示与交换分开，同时采用专门的信息交换设备来提高性能。信息交换应该具有充分的灵活性，可根据实际需要组建多节点交换网络以提高总体交换性能。另外，信息在传输过程中需要通过压缩来提高传输速度。

⑥可管理性问题：提供日志、审计、会话管理、传输优先级等。

Web Services技术是解决复杂网络环境下消息传递的有效技术。但是Web Services技术的功能不仅仅是解决消息传递问题。Web Services是一种在Internet上共享数据和功能的手段，它的调用应当

遵循特定的技术标准，如 XML 协议、简单对象访问协议 SOAP、服务描述协议 WSDL、服务注册和查找协议 UDDI，Web Services 为组织间的交互提供了一种标准“接口”方式。

Web Services 是一种新型的软件应用，能够通过 XML 消息及 Internet 协议完成与其他软件应用的直接交互。Web Services 的目的和作用是提供一种国际统一的规范和技术，进行 Internet 上各种软件应用的统一功能描述和功能共享，为功能整合集成和信息交换处理提供实现基础。XML 提供了在不同平台/系统之间的数据层集成能力，Web Services 提供了一种在不同平台/系统之间在软件应用层进行功能自动整合集成和自动化处理所需要的技术架构。

Web 服务技术建立在 XML 技术的基础上，其功能接口及调用形式可以通过 XML 标准定义、描述和检索，Web 服务的体系架构与 Web 应用的 N 层架构是类似的，不同点在于最上层的面向浏览器的 Web Server 被面向程序（Web Services Client）的 Web 服务所取代。而使用 Web 服务的程序可以是桌面应用程序，同样也可以是另一个 Web 服务。构筑 Web 服务的 Web 服务技术家族的主要成员有 XML Schema、SOAP、WSDL 和 UDDI，它们都是完全基于 XML 技术的。其中 XML Schema 为在不同系统（Web 服务）之间交换数据而提供了一个核心的跨平台数据建模工具；SOAP 为在不同系统之间实施与平台无关的信息交互定义了一套基本的元规则和跨平台消息机制，SOAP 是 Web 服务体系中服务交互的基础架构；WSDL 则是 Web 服务接口界面的跨平台描述工具，依靠 WSDL，Web 服务的交互过程就能被系统自动处理；UDDI 则是解决动态服务集成问题的一种尝试，实现基于 Web 的信息与服务自动注册、查找和访问。Web Services 技术使得底层平台对应用交互透明，应用的互操作能力得到

了前所未有的提升，从而成为新一代 Internet 软件技术。

从使用者的角度来看，Web 服务是一种部署在 Web 上的对象/组件，它具备以下特征：

（1）完好的封装性

Web 服务作为是一种部署在 Web 上的对象，具备对象的良好封装性，对于使用者而言，能且仅能查找、定位和使用该服务提供的功能列表及功能实现。

（2）松散耦合

这一特征源于对象/组件技术，当一个 Web 服务的实现发生变更的时候，调用者是不会感到这一点的，对于调用者来说，只要 Web 服务的调用接口不变，Web 服务实现的任何变更对用户来说都是透明的，甚至是当 Web 服务的实现平台从 J2EE 迁移到了 NET 或者是相反的迁移流程，用户都可以对此一无所知。对于松散耦合而言，尤其是在 Internet 环境下的 Web 服务，需要有一种适合 Internet 环境的消息交换协议，而 XML/SOAP 是目前最为适合的消息交换协议。

（3）使用协议的规范性

首先，作为 Web 服务，对象界面所提供的功能应当使用标准的描述语言来描述（如 WSDL）；其次，由标准描述语言描述的服务界面应当是能够被发现的，因此这一描述文档需要被存储在私有的或公共的注册库里面；使用标准描述语言描述的协议规范将不仅仅是服务界面，它将被延伸到 Web 服务的聚合、跨 Web 服务的事务、工作流等，而这些又都需要服务质量（Quality of Service）机制的保障；最后，因为安全机制对于松散耦合的对象环境的重要性，所以需要对诸如授权认证、数据完整性（如签名机制）、消息源认证以及事

务的不可否认性等运用规范的方法来描述、传输和交换。

（4）标准协议规范的开放性

作为 Web 服务，需要使用开放的标准协议进行描述、传输和交换，这些标准协议具有完全免费的规范，以便由任意方进行实现。一般而言，绝大多数规范将最终由 W3C（World Wide Web Consortium，万维网联盟）作为最终版本的发布方和维护方。

（5）高度的可集成性

由于 Web 服务采取简单的、易理解的标准 Web Services 协议作为组件界面描述和协同描述规范，屏蔽了不同软件平台的差异，无论是 CORBA、DCOM 还是 EJB（Enterprise JavaBean，JavaEE，服务器端组件模型）都可以通过这一种标准的协议进行互操作，实现了在当前环境下最高的可集成性。

综合当今的 Web 应用以及 Web 服务的特点，Web Services 的应用领域可以分为 4 类：

①面向业务的 Web 服务（Business-Oriented Web Services）：该类服务针对的是面向企业级应用的服务，如企业内部的 ERP（Enterprise Resource Planning，企业资源计划）系统，企业间的 SCM（Supply Chain Management，供应链管理）、CRM（Customer Relationship Management，客户关系管理）等系统。当这些系统以 Web 服务的形式在网络（Internet 和 intranet）中出现时，企业内的应用集成将更加容易，而在企业间的众多合作伙伴的系统对接也将不再是无法完成的任务。

②面向客户的 Web 服务（Consumer-Oriented Web Services）：此类服务针对的是 B2C（Business to Customer，商户对客户）应用的改造，为这些 Browser-Oriented 的 Web 应用增加了 Web 服务的应用

界面，使得第三方的桌面工具或其自身提供的增值桌面工具能够利用更优秀的用户界面提供跨越多个 B2C 服务的桌面服务。这将使得用户使用 Internet 更为方便，能够获得更加便捷的服务。比如完全可以在个人理财桌面系统中集成（调用）Internet 上的股票价格查询 Web 服务、机票预定 Web 服务等，使得个人理财应用的自动化程度更高。

③面向设备的 Web 服务（Device-Oriented Web Services）：此类服务的使用终端一般是手持设备和日用家电，对于前者而言，可以在不需修改原有网络服务的体系架构前提下，令先前的网络服务支持除 PC 以外的各种终端，比如 Palm、PocketPC、手机等。这里一些类似天气预报服务、E-mail 服务、主动信息服务等将变得更为有效和便捷。而对于日用家电，有了 Web 服务作为基础框架，智能型的日用家电将真正获得标准的支持，从而具有广泛使用的可能。

④面向系统的 Web 服务（System-Oriented Web Services）：一些传统意义上的系统服务，比如用户权限认证、系统监控等，如果被迁移到全球范围的 Internet 上，或者企业内部的 intranet 上，其作用范围将从单个系统或局部网络拓展到整个企业网络或整个 Internet。如此，基于同一系统服务的不同应用将得以在整个 Internet 环境中部署，譬如跨国企业的所有在线服务可以使用同一个用户权限认证 Web 服务。

1.2.3 消息中间件

中间件是在分布式系统中用来管理其固有的复杂性和异构性的一类软件，它位于操作系统之上和应用程序之下，提供一种通用的较高层的程序抽象。中间件提供的服务具有标准的程序接口和协议，

针对不同的操作系统和硬件平台，它们可以通过符合接口和协议规范等多种实现形式。

通常意义下中间件具有以下一些特点：满足大量应用的需要；运行于多种硬件和操作系统平台；支持分布式计算，提供跨网络、硬件和操作系统平台的透明的应用或服务交互；支持标准的协议；支持标准的接口。由于标准接口对于可移植性和标准协议对于互操作性的重要性，中间件已成为许多标准化工作的主要部分。对于应用软件开发，中间件远比操作系统和网络服务更为重要，中间件提供的程序接口定义了一个相对稳定的高层应用环境，不管底层的计算机硬件和系统软件怎样更新换代，只要将中间件升级更新，并保持中间件对外的接口定义不变，应用软件几乎不需任何修改，从而保护了企业在应用软件开发和维护中的重大投资。

随着计算机软件技术的发展，中间件技术也已经日渐成熟，并且出现了不同层次、不同类型的中间件产品。中间件的分类方式很多，按照 IDC（Internet Data Center，互联网数据中心）的分类方法，中间件可分为六类，分别是终端仿真/屏幕转换中间件、数据访问中间件、远程过程调用中间件、消息中间件、交易中间件和对象中间件。

消息中间件（Message Oriented Middleware，MOM）指的是利用高效可靠的消息传递机制进行平台无关的数据交流，并基于数据通信来进行分布式系统的集成。通过提供消息传递和消息排队模型，它可在分布环境下扩展进程间的通信，并支持多通信协议、语言、应用程序、硬件和软件平台。应用级的会话有同步和异步两种方式，消息中间件可以既支持同步方式，又支持异步方式，实际上它是一种点到点的机制，因而可以很好地适用于面向对象的编程方式。

消息传递和排队技术有以下 3 个主要特点：

①通信程序可在不同的时间运行：程序不在网络上直接相互通话，而是间接地将消息放入消息队列。因为程序间没有直接的联系，所以它们不必同时运行。消息放入适当的队列时，目标程序甚至根本不需要正在运行；即使目标程序在运行，也不意味着要立即处理该消息。

②对应用程序的结构没有约束：在复杂的应用场合中，通信程序之间不仅可以是一对一的关系，它们还可以进行一对多和多对一方式甚至上述多种方式组合的通信。多种通信方式的构造并没有增加应用程序的复杂性。

③程序与网络复杂性相隔离：程序将消息放入消息队列或从消息队列中取出消息来进行通信，与此关联的全部活动（比如维护消息队列、维护程序和队列之间的关系、处理网络的重新启动和在网络中移动消息等）由 MOM 处理，程序不直接与其他程序通信，从而避免涉及网络通信的复杂性。

1.2.4 国内外的数据交换平台产品现状

1.2.4.1 国外的数据交换平台产品现状

当前许多国外知名的大公司进军数据交换领域，在这方面已经出现了一批成熟的产品。

IBM 提出了 Web Sphere Information Integrator 数据整合方案，提供了信息集成框架的基础，有助于客户实时访问、操作和集成各种不同的分布式数据。方案中的每一个产品都能使客户从各种不同的分布式数据和内容源抽象出公共数据模型，并使客户能够将它们当

作单一源进行访问和操作。每个产品都支持一个用户社区，用户社区主要是根据其成员可以访问的数据和他们支持的开发社区定义的。

Microsoft 提出了 BizTalk 框架，它以 XML 的形式表达商业文档和数据消息。应用系统间的数据交换采用基于 BizTalk 框架格式化的 XML 消息，而不需要考虑编程语言、网络协议、数据库或操作系统，应用系统仅仅需要能够格式化、传输、接收和使用标准化 XML 消息。由个人或组织向 BizTalk.Org 网站递送其商业数据规范，如果该规范通过了认证测试，作为公用规范发布，任何个人或组织都可以从 BizTalk.Org 网站上下载免费使用该 XML 规范。该框架以一致的形式建立大量的大纲，为应用工具和基础软件供应商提供了明确的设计目标，便于用户开发企业应用集成产品。

Sybase 提出了 Sybase DXP 数据交换平台，Sybase DXP 为电子政务系统内及系统间的信息交互和共享提供了一个集成化数据交互和共享空间，具有较好的开放性，可同时满足用户各类操作系统平台、数据源及应用系统间的数据交换需求。Sybase DXP 根据企业的网络现状和业务需求，提出了四种解决方案：大数据量实时传输 DXP 解决方案，低带宽数据传输 DXP 解决方案，异构数据库数据传输 DXP 解决方案，企业间数据交换 DXP 解决方案。

1.2.4.2　国内的数据交换平台产品现状

许多国内公司也推出了一些数据交换平台产品。例如：

鼎天软件提出了基于 J2EE 和 XML 技术的数据交换平台，根据政府行业的应用特点，在通用的消息中间件上封装面向应用系统的功能软件，呈现给用户的是一组 API 接口。帮助用户建立统一的数据传输和数据交换规范，实现异构信息的高度共享和综合利用。鼎

天数据交换平台是一个基于标准的开放平台，主要运用于政府信息化建设方面。在电子政务应用实施过程中，信息化环境错综复杂，不同单位、不同部门的应用系统不同，而这些应用系统之间经常要交换数据。鼎天数据交换平台提供在不同的系统、不同的数据源、不同的运行环境间进行数据交换的功能。

北大方正技术研究院基于J2EE体系结构，推出了面向信息资源整合的跨地域、跨部门应用技术框架，为横跨各政府机构的服务、监管职能的业务实现，为同一机构内多个部门不同业务系统之间的数据整合和协同办公，提供了进行有效转换和交流的数据交换平台——方正汇通。

福建省凯特科技有限公司推出的基于.Net开发，以XML、Web Services为基本技术手段，以实现不同应用系统的数据交换、共享和集成为目标的凯特SecExchange数据交换平台。同时，SecExchange通过Web Services提供开发接口，各个应用系统可通过这些接口在企业内部实现数据交换功能。

中创软件推出的“电子政务数据交换平台解决方案”，是基于中创软件Infor系列中间件技术，结合政府信息化建设现状及发展需求而推出的，使得各政府部门之间的基础数据共享，让基础数据发挥更大的社会价值，使政府从宏观上把握经济运行的整体情况。该方案主要实现：实现政府部门之间数据的安全、可靠交换和共享，避免数据重复采集，保持各部门基础数据的一致；实现数据的即时整合，并对全局数据进行灵活的多维度分析和多样式展示，为管理层监控和决策提供有效支持。

北京中关村科技软件有限公司研发了CenDXS数据交换平台，它遵循标准的、面向服务架构（Service Oriented Architecture，SOA）

的方式，基于企业服务总线（Enterprise Service Bus，ESB）技术，采用 XML 技术标准和规范，为不同应用系统、不同数据库之间的互联互通提供包含提取、转换、传输等操作的数据整合服务，实现扩展性良好的“松耦合”结构的应用和数据集成；它使用分布式部署和集中式管理架构，有效解决各节点之间的数据交互，在安全、方便、可靠地进行信息交换的同时保证数据的一致性和准确性，实现了数据的一次采集、多应用系统共享，为各种应用和决策支持系统提供良好的数据环境。

通过对 Web 应用中大规模数据交换的相关技术及相关产品的分析，可以看出在 Web 应用中，大规模数据交换具有以下特点：

①异构性：应用系统的实现可能基于多种平台，系统的消息传输可能是同步模式，也可能是异步模式，信息资源在形式上没有统一的体系和结构，处于非结构化的状况，资源的存储形式也有所不同，如文件系统、文档数据库、关系型数据库、媒体型等。

②多样性：在 Web 应用中，交换的数据可能是文本类型的数据，也可能是图片、视频、音频等其他类型的文件，而交换数据的大小也存在巨大的差异，可能只有几个字节，也可能是数十兆甚至上百兆数据量的数据文件。

③数据传输的可靠性：由于数据交换应用广泛，在一个信息交换网络中，无论是系统还是网络都可能存在异常、故障以及其他风险问题，从而容易造成数据丢失、传输不完整等问题，因此要求数据交换具有很高的可靠性，尤其是在商业贸易、政府办公等电子商务、电子政务领域。

④数据传输效率：基于 Web 服务进行数据交换，尽管 SOAP 提供了强大的互操作性，但是其自身特点决定了其传输效率较低，将

数据放入 SOAP 消息中进行传输时，在编码转换以及报文构造和解析上需要消耗大量时间，对于数据传输效率要求较高或者大数据量传输的应用，SOAP 不能很好地满足实际需要。

目前已经有众多的组织和公司开始关注数据交换领域，并且相继推出了一些数据交换平台产品，这些产品能够为不同的系统、不同的运行环境提供数据交换的功能。但是，这些产品一方面与项目本身耦合紧密，无法适应广域网下电子政务、电子商务这些 Web 应用的数据交换需求，比如这些产品通常都采用单一的传输协议，没有制订一套完善独立的数据传输机制；另一方面数据交换除了需要提供基础消息服务之外，还需要保证数据交换可靠性、数据交换传输效率，并且需要对公共数据信息的存储、发布及发现进行支持。因此，为了解决信息化过程中数据资源的互操作问题，需要分析数据交换的模式及基础服务，并进一步研究大规模数据交换中的相关关键技术，提出解决方案，研制数据交换平台原型系统，并在实践应用中加以验证。

1.3 数据交换关键技术

本书基于 Web 服务技术，分析大规模数据交换所面临的问题，建立基于 Web 服务的数据交换平台框架，解决其中的若干关键技术，保证数据交换传输的可靠性以及数据交换传输的效率，研究基于分布式存储模型的公共数据服务中心，并且研制数据交换平台原型系统。本书研究的数据交换关键技术主要包括以下几个方面：

①基于 Web 服务的数据交换平台框架：数据交换技术主要关注数据是如何在发送方、中间环节、接收方之间实现数据通信的。数

据交换平台是不同应用之间进行数据传输的基础，从而达到不同应用之间进行资源共享互通的效果。对使用 Web 服务进行数据交换的必要性进行分析，并分析数据交换的应用模式。在此基础上提出基于 Web 服务的数据交换平台框架，并由此提出基于 Web 服务的数据交换的相关关键技术。

②基于 Web 服务的数据交换的可靠性：参与数据交换的实体来自不同自治域，管理结构不同，资源状态的不可控，使得数据交换的可靠性成为网络环境下动态复杂应用的核心问题之一。在对消息传输的可靠性进行分析的基础之上，从数据交换消息层定义了消息可靠性，讨论了消息可靠性的参数，分析数据交换的不可靠性因素，并提出一种可靠的消息传输机制，解决消息传输过程中可能出现的消息确认、消息重复、消息丢失等问题。

③基于 Web 服务的数据交换的效率：数据交换可能发生在任何两个应用系统之间，交换双方可能处于同一个局域网内，也可能处于广域网上相距甚远的两个网络节点上，交换过程可能跨越多个网络自治域，不同的应用及不同的网络都会对数据交换的效率产生一些影响。数据交换基于 Web 服务，Web 服务本身也会对数据交换的效率产生影响。需要分析 Web 服务传输性能，指出影响 Web 服务性能的因素，提出 Web 服务性能优化方案，并在此基础上提出基于 Web 服务的类型映射机制和基于滑动窗口的 SOAP 消息传输机制，设计拥塞控制算法，用于优化基于 Web 服务的数据交换的效率。

④基于分布式信息存储模型的数据服务中心：在数据交换系统中，信息资源分布在不同的节点中来管理和维护，数据服务中心的主要目标是向用户提供信息共享服务，从而更好地对资源信息进行描述、方便资源的发布与发现。在对数据交换中数据服务中的可扩

展性、发现效率和有效性进行分析的基础上，设计了一个分布式的信息存储模型，讨论了注册中心的组织，提出了信息动态注册与更新机制，设计了一个分布式信息搜索算法，并设计实现了基于UDDI的数据服务注册中心。

⑤设计并实现基于Web服务的数据交换平台：提出一种数据交换协议，在此基础之上设计实现了基于Web服务的数据交换平台，介绍了其总体结构，阐述了其中的主要模块和关键机制如数据交换引擎、适配器及事务机制等，并讨论了数据交换平台在电子政务中的应用部署。

第 2 章
基于 Web 服务的数据交换平台框架

数据交换技术主要关注数据是如何在发送方、中间环节、接收方之间实现数据通信的，而数据交换平台框架则是不同应用之间进行数据传输的基础，从而达到不同应用之间进行资源共享互通的效果。本章首先分析了数据交换的应用背景及功能，又分析了基于 Web 服务进行数据交换的必要性，然后针对数据交换的应用模式进行了分析，并在此基础上提出了基于 Web 服务的数据交换平台框架。

2.1 数据交换与 Web 服务

2.1.1 数据交换概述

2.1.1.1 数据交换应用分析

随着信息技术在世界范围内的迅猛发展，特别是互联网技术的普及应用，电子政务的发展正在成为当代信息化的最重要的领域之一。当前，我国大规模的国家电子政务建设已经蓬勃兴起，电子政务已经成为我国信息化发展战略中的核心，各级政府也将电子政务

作为政务体制改革、提高管理效能、改善政府形象的重要手段。但是，在各地方各部门的电子政务建设过程中，由于缺乏统一的规划和标准，造成了现有的各种电子政务信息系统没有规范和标准可以遵循，每个系统都相对独立和分散。系统之间的异构性和封闭性造成了系统之间不能信息共享，形成了一个个的信息孤岛。而电子政务未来的发展目标是整合和优化政府的资源，实现全面的信息共享，在此基础上实现政府各部门职能的整合、跨部门的协同业务流程和协同办公，提高信息共享和业务协同的范围和程度。这就需要有一个统一的数据交换基础平台来支持各种异构平台之间的信息和数据交换。

数据交换是网络环境下的分布式应用的共性、基础性和关键性的需求。在电子政务领域，数据交换的作用显得尤为突出。因为，政务信息资源共享和政府业务集成是电子政务主要和重要的内容，无论是信息资源共享和业务互操作，这一切都必须以数据交换作为基础。

数据交换基础平台作为其他业务系统的底层平台为各业务系统提供数据的标准化机制和统一的数据传输机制。其他业务系统基于这样的平台相互连接，实现数据交换和信息共享。这种多部门、多系统的相互连接和整合，形成了一个处于底层电子政务基础网络上的数据交换逻辑网络。在这个网络中，无论是在同一系统的内部还是在不同的应用系统之间，都存在着大量的数据交换需求，如电子公文的分发、政务数据的共享与传输等。在此过程中产生了许多技术上的问题，如数据交换的可靠传输机制、数据传输的效率、公用数据信息的存储等，这些问题都是当前面向分布式应用的数据交换主要的研究热点之一。

2.1.1.2　数据交换功能分析

数据交换就是解决信息化过程中数据资源的问题即解决不同的异构系统之间的数据资源的整合，从而实现在不同的系统之间数据的共享。数据交换中心是实现异构系统之间、新老系统之间信息透明交换的一种解决方式。通过采用统一的数据交换标准，各应用系统与数据交换中心相连，通过数据交换中心来实现数据共享和路由，由于隔离了数据存储层和应用层，使得应用与其底层的数据结构和存储方式无关，从而不需要对原有业务系统进行改造，也不需要对已有的业务流程重新开发。这种连接方式实现了数据的无缝交换和共享访问，保证了各业务系统的有效协同，同时又能保证各应用系统的相互独立性和低耦合性，从整体上提高了系统运作效率和安全性。

数据交换是数据交换系统的基本功能，可实现系统内任意合法用户之间的数据安全传输。数据交换是各类应用系统共享信息资源的公共渠道。

数据交换功能的具体需求如下：

- 支持异构数据的表示。基于数据源分布的多样性，以及数据类型的多样性，数据交换和数据描述必须以 XML 为基础。
- 支持异构应用系统之间的数据交换：为了实现穿越防火墙、跨平台的数据交换，数据交换应该使用 SOAP 作为安全通信的基础。
- 支持结构化数据（如关系数据库数据）和非结构化数据（如图形文件）的交换。

- 支持“1对1”和“1对多”的数据传输模式。
- 支持自动路由选择。
- 提供断点续传机制。
- 提供可靠传输机制。
- 提供完善的安全机制：为适应不同的数据交换和传输安全需求，数据交换应该提供数据内容的安全机制、端到端的传输安全机制、身份认证和授权管理机制。

2.1.2 Web服务分析

Web Services是一种新型的软件应用，能够通过XML消息及Internet协议完成与其他软件应用的直接交互。Web Services的目的和作用是提供一种国际统一的规范和技术，进行Internet上各种软件应用的统一功能描述和功能共享，为功能整合集成和信息交换处理提供实现基础。

面向服务的体系结构（Service Oriented Architecture，SOA）是在分布式环境下建立松耦合软件系统的一种典型的体系结构（见图1），提供了按功能组织服务的模式，每个服务都提供一组定义良好的功能集合，而基于SOA的应用可被视为一组相互交互的服务，这种基于服务的形式功能描述为系统的灵活性、可扩展性和开放性提供了基础，有效地支持系统透明性、动态升级和演化。

①服务请求者：即服务的消费者，根据所要执行的功能选择并定位服务。它可以是一个独立的客户端代理，也可作为服务提供者向其他服务请求者提供高层次的服务。

②服务提供者：为服务请求者提供符合服务的语法和语义约定的服务，并按照指定的语法格式将服务描述信息注册到服务注册表。

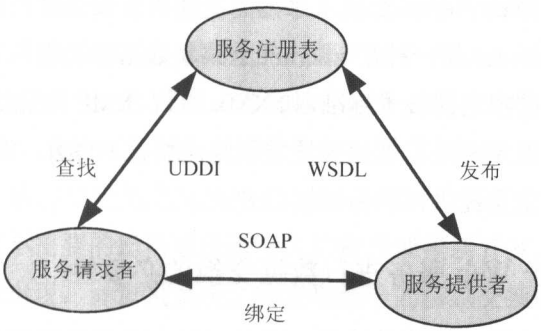

图 1　面向服务的体系结构

③服务注册表：作为一类特殊的服务，遵循约定的服务接口，存储由服务提供者所发布的服务描述信息，执行服务请求者提出的服务发现请求。对于一个规模较大的分布式系统，由于所提供的服务种类与数量众多，往往需要多个注册表服务形成一定的拓扑结构并按照一定的信息交换和管理协议进行协同工作。

在面向服务的体系结构中，服务请求者与服务提供者之间的“松耦合绑定关系”是关键，即请求者在运行时通过与服务注册表进行交互以获得可用的服务提供者，并动态确定与哪个服务提供者所提供的服务做运行时绑定。在分布计算系统中，服务的“高内聚”和“低耦合”特征确保了面向服务体系结构的这种松耦合关系的可行性。在具体的系统实现中，服务可以有不同的表现形式，如 CORBA 系统中的 CORBA 对象、JINI 系统中符合 RMI 接口规范的 Java 远程对象、Web 服务环境中的 Web 服务，它们都可以用来构建面向服务的体系结构。

Web Services 技术序列是解决复杂网络环境下消息传递的有效

技术。但是 Web Services 技术序列的功能不仅仅是解决消息传递问题。Web Services 是一种在 Internet 上共享数据和功能的手段，它的调用应当遵循特定的技术标准，如 XML 协议、XML 消息协议 SOAP、服务描述协议 WSDL、通用描述发现集成协议 UDDI，Web Services 为组织间的交互提供一种标准接口方式。

2.1.3 基于 Web 服务进行数据交换的必要性

从本质上讲，数据交换是一种可以独立于具体应用业务逻辑的共性行为，对应用系统业务逻辑产生直接影响的是交换数据的语义，而数据交换只关注交换规则的定义和交换过程的实现，正如邮政传递系统，需要制定的是邮政编码、地址表示、邮寄费用、传递路径等一系列规则，并构建完善的传递体系以保证邮件的安全达到，至于邮件本身不是邮政系统关心的内容。

在数据交换中，数据的封装格式以及传输模式是需要进行关注的，它们的解决方式会影响到数据交换的效率以及相应的数据交换网络。例如，一个城市的政府机关达几十个甚至 100 多个，它们之间的数据交换呈网状交换。很明显，以局的信息系统（即所谓信息孤岛）为单位设计同其他局的信息交换，其代价、技术的复杂度是相当高的。这种集成的方式是用手工编程，对每一个单位都要开发一个接口。如果有 N 个应用系统需要集成，就需要建立 $N(N-1)/2$ 个接口程序。如果有多个部门，其接口程序的开发工作量是相当可观的。如果某个单位的信息系统发生变化，则相应的接口都会要进行调整。这样每年花费在应用接口上的维护费也是相当可观的。

而 Web 服务的目的和作用是提供一种国际统一的规范和技术，进行 Internet 上各种软件应用的统一功能描述和功能共享，为功能整

合集成和信息交换处理提供实现基础。通过使用 Web 服务技术，所有的应用数据的封装都是标准化的，都是基于 SOAP 报文格式进行封装，并且使用 SOAP 消息进行传输；每个应用程序的接口描述都是使用 WSDL 来进行描述，公用信息使用通用描述发现集成协议来进行存储，从而使应用之间的维护、安全控制、信息过滤都变得非常容易。它不需要的遗留系统进行大的修改或重新开发。由于 Web 服务技术的出现，计算机系统之间实现统一的信息交换成为可能。

2.2　数据交换应用模式分析

数据交换是各类网络应用的共性、基础性和关键性的功能需求。在对数据交换应用模式进行分析之前，先给出如下数据交换网络的定义。

假定 V 是数据交换应用层交换网络结点的集合，E_1 是 V 之间连接，则应用层数据交换网可以简单地表示为图 $G_1=(V, E_1)$。

假定 v_1，v_2 分别表示进行数据交换的两个终端节点，d 是交换的数据，则应用层数据交换网络之上的数据交换 Ex_1 则可简单的表示为 $Ex_1=(G_1, v_1, v_2, d)$。

在数据交换网络 G_1 中，所有终端节点均连接到所在自治系统的路由器上，终端节点之间没有直接连接，v_1 和 v_2 是终端节点，即 $v_1 \in V$，$v_2 \in V$，在一个包含数千个节点的广域网络内，v_1 和 v_2 之间的数据交换可能需要跨越多个网络自治域，数据报文的传递需要经过若干个路由器，才最终传送到目标节点的应用系统。由于除了节点 v_1 和 v_2 之外，数据报文经过的中间路由节点都不可能了解应用层的语义，因此，数据报文在传递过程中的路径选择实际上与应用毫

不相关。在交换网络中，所有节点均为参加信息交换应用的终端节点，这些节点实际上都是运行在具体计算机上的应用系统，所有节点都能理解应用的功能和交换的语义。因此，对数据交换的应用模式进行分析是必要的。

2.2.1 部署结构

数据交换平台是不同应用之间进行数据交换的平台，可以实现多个应用之间的数据交换。数据交换平台通过建立统一的信息交换模式，对各个已有或将有的应用系统包括办公自动化系统、关系型数据库应用系统及其他应用系统中的数据进行交换和处理，将数据传输到另一个系统中去再利用，从而消除信息孤岛问题；或者控制公文、数据的流转和传输，实现协同办公和并联审批；或者在此基础上将数据以统一的格式在门户上对外发布。

数据交换平台可以有两种结构：一种是两两直接连接的网状结构；另一种是建立交换中心的星形结构。

网状结构（见图 2）在需要交换数据的节点很少的情况下是适用的，但是当节点数量（N）增加之后，由于每个节点所支持的交换协议难以统一，任意两个节点之间的交换协议以及数据格式不尽相同，因此系统的复杂度呈指数增长。而且在这种结构下，每个节点都需要记录其余 N–1 个节点数据交换软件的部署位置，一旦增加、减少或者改变一个节点的部署位置都需要通知所有其余 N–1 个节点，缺乏灵活性，不利于扩展。

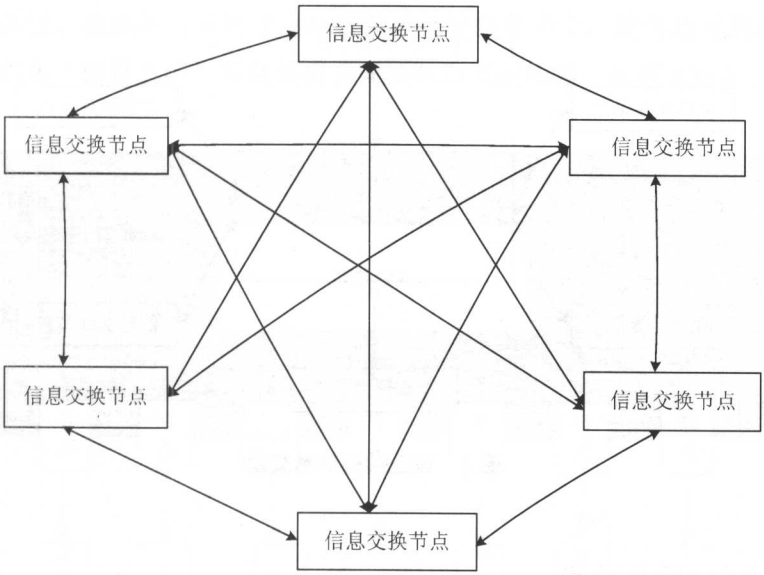

图 2　网状结构信息交换

星型结构的数据交换平台的体系结构如图 3 所示。从图中可以看出，数据交换平台可以分为信息交换中心和应用信息交换节点两大部分，信息交换中心部署了数据交换中心系统；信息交换节点连接到信息交换中心，节点内部的数据库、应用系统等通过节点的适配器连接到节点。

相比之下，星型结构的数据交换平台就不存在这些缺点。数据交换平台采用星型模式，利用该结构，可以统一数据交换协议、降低结构复杂度；可以在交换中心定义数据格式标准，实现数据格式和内容的转换；可以统一应用节点的地址，实现地址的解析和路由；可以在交换中心增加日志、统计、管理和监控功能；可以实现对数据的任意加工和处理等。

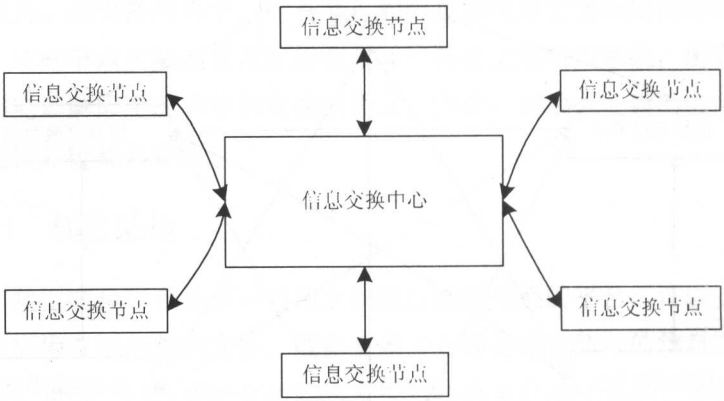

图3 星型结构信息交换

2.2.2 信息总线

在星型结构下，某个节点的应用系统需要数据时，只需向中心发送请求，中心就会到其他节点提取数据给该节点；节点的应用系统有数据需要发送时，也可以把数据发送给中心，由中心把数据转发给需要数据的其他节点。该节点不必关心目标节点的具体位置以及目标节点的系统是否启用。同样地，当节点的应用系统接收到数据时，也不必关心数据的真实来源以及数据原来的物理形态，数据可以来自中心数据库，或是来自中心的协同办公系统，或是来自其他的一个甚至多个节点。

交换中心的整体行为就像一个虚拟的中心数据库，同时又像一个交换机。整个数据共享和交换的底层实现和存储机制对各应用节点是透明的。从节点应用系统的角度来看，信息交换中心和所有的信息交换节点在逻辑上构成了一个“信息总线”，节点内部的各种应

用系统、数据库只要通过适配器挂接到信息总线上，就能透明地通过这条“信息总线”实现数据的交换和公文的传输，如图 4 所示。

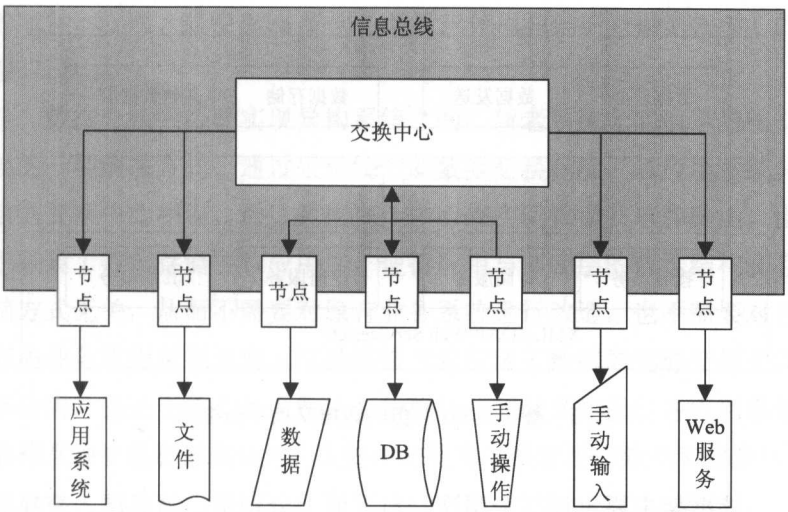

图 4　信息总线

2.3　基于 Web 服务的数据交换平台框架

2.3.1　总体设计

基于数据源分布的多样性、数据结构和类型的多样性，要求数据交换平台以 XML 和基于 XML 的消息机制为基础，同时结合各类数据提取、转换适配器，实现基础的数据表示和数据交换。以此为基础，提出了基于 Web 服务的数据交换平台框架，如图 5 所示。

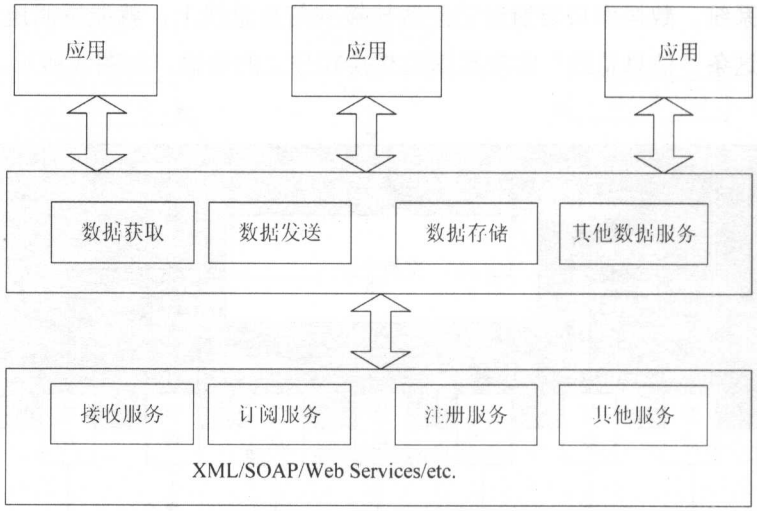

图 5　基于 Web 服务的数据交换平台框架

该框架主要包括三个层次，自下而上分别是：消息层、交换层和应用层。

①消息层：支持 XML、SOAP 和 Web Services 等技术，为数据交换提供基于 XML 的消息机制和 Web 服务的运行环境。消息层构建于物理网络之上，为数据交换提供基础的消息服务，包括接收服务、订阅服务以及注册服务等。

②数据交换层：利用 XML 表示数据，用 SOAP 作为通信机制，利用 Web 服务技术连接节点内部的系统。提供完善的数据交换协议，支持数据推拉、数据路由、数据验证、数据管理、数据存储等数据交换服务。它是整个数据交换平台的核心，负责完成数据交换相关的所有功能，它主要由以下几个部分组成：负责数据存储的中心数据库，数据获取服务、数据发送服务以及交换中心的核心程序。其

中又以核心程序最为重要，它负责数据的传输、路由、安全以及转换处理，从而实现数据交换的核心功能。

③应用层：是数据交换中心的上层应用，包括对数据交换中心的配置、管理、监控 Web 界面，是用户使用和维护数据交换中心的接口。

数据交换中心是实现异构系统之间、新老系统之间信息透明交换的一种解决方式。通过采用统一的数据交换标准，各应用系统与数据交换中心相连，通过数据交换中心来实现数据共享和路由，由于隔离了数据存储层和应用层，使得应用与其底层的数据结构和存储方式无关，从而不需要对原有业务系统进行改造，也不需要对已有的业务流程重新开发。这种连接方式实现了数据的无缝交换和共享访问，保证了各业务系统的有效协同，同时又能保证各应用系统的相互独立性和低耦合性，从整体上提高了系统运作效率和安全性。

2.3.2　数据交换功能

数据交换功能是指数据交换中心利用 SOAP 协议实现数据交换，它是数据交换中心的核心功能，它又可以分为如下几个方面：

①数据可靠的传输：要求传输的数据不受各种网络因素的影响，准确无误地到达接收方，不能有丢失、重复、损坏。同时还要能防止有人恶意的破坏传输，包括篡改、监听、冒名发送、重放发送、抵赖等手段。数据传输应支持“推”“拉”两种数据传输方式，所谓“推”，是指发送方主动向接收方发送数据，所谓“拉”，是指接收方向发送方索取数据。

②数据传输控制：数据的传输控制包括对数据拥塞、数据流量和数据优先级的控制，以及需要在网络异常情况下能进行断点续传，

从而提高传输效率。解决上述问题要求在实现数据传输时，要增加数据切片、多级收发队列和收发缓冲池等机制。

③数据的路由转发：交换中心必须能对数据传输的目的地址进行解析，以实现数据的路由转发。

④公共数据的存储及共享：交换中心需要对一些公共的信息进行存储，并提供数据的发布和发现机制，从而能够让系统的传输更具有效率。

2.4 本章小结

数据交换中心是实现异构系统之间、新老系统之间信息透明交换的一种解决方式。通过采用统一的数据交换标准，各应用系统与数据交换中心相连，通过数据交换中心来实现数据共享和路由。本章首先对数据交换的应用背景及功能进行了分析，又讨论了基于Web 服务进行数据交换的必要性，然后针对数据交换的应用模式进行了分析，在此基础上提出了基于 Web 服务的数据交换平台框架。

第 3 章
数据交换可靠性研究

参与数据交换的实体来自不同自治域，管理结构不同，资源状态的不可控，使得数据交换的可靠性成为网络环境下动态复杂应用的核心问题之一。本章在对消息传输技术的可靠性进行分析的基础上，从数据交换消息层定义了消息可靠性，讨论了消息可靠性的参数，对基于 Web 服务的数据交换框架的不可靠因素进行了分析，主要针对消息确认、消息重发、消息丢弃等方面，设计了可靠的数据交换消息机制，并讨论了其实现的一些技术，从而满足数据交换中的可靠消息传递需求。

3.1 消息传输的可靠性分析

3.1.1 传统的 TCP 协议的可靠性

传统的 TCP（Transmission Control Protocol，传输控制协议）处理消息可靠性问题一般采取消息确认、超时重传、流量控制和拥塞控制等措施。应用数据被分割成 TCP 认为最适合发送的数据块。由 TCP 传递给 IP 的信息单位称为报文段或段。当 TCP 发出一个段

后，它启动一个定时器，等待目的端确认收到这个报文段。如果不能及时收到一个确认，将重发这个报文段。当 TCP 收到发自 TCP 连接另一端的数据，它将发送一个确认。这个确认不是立即发送，通常将推迟几分之一秒。TCP 将保持它首部和数据的检验和。这是一个端到端的检验和，目的是检测数据在传输过程中的任何变化。如果接收端的检验和有差错，TCP 将丢弃这个报文段和不确认收到此报文段。既然 IP 数据报文会发生重传，TCP 的接收端必须丢弃重复的数据。TCP 还能提供流量控制。TCP 连接的每一方都有固定大小的缓冲空间，TCP 的接收端只允许另一端发送接收端缓冲区所能接纳的数据。这将防止因处理速度较快的主机导致较慢主机的缓冲区溢出。

3.1.2 JMS 消息队列可靠性

JMS（Java Message Service，Java 消息服务）是由 Sun Microsystems 开发的一种与厂商无关的 API（Application Programming Interface，应用程序编程接口），JMS 定义了一组在 Java 企业系统中传递消息的规则，并且声明了一些方便应用组件和消息传递系统之间的消息交换的接口，同时也能实现各种消息系统之间的互操作和无缝链接。它提供创建、发送、接收、读取消息的服务，通过定义一组公共的应用程序接口和相应语法，使得 Java 应用能够和各种消息中间件进行通信，这些消息中间件包含 IBM MQ-Series、Microsoft MSM 等。通过使用 JMS API，开发人员无须掌握不同消息产品的使用方法，便可以使用统一的 JMS API 来操纵各种消息中间件，通过使用 JMS，能够最大限度地提高消息应用的可移植性。

JMS 支持两种基本的消息传递机制。第一种机制是点到点的消

息传递，在点到点模型中，通常一个客户端与一个消息队列进行绑定。在特殊情况下，一个客户端可以与多个队列进行绑定。如客户端 A 要与客户端 B 进行通信，客户端 A 不需要知道客户端 B 是否在线或者是否能与客户端 B 建立连接，只需知道客户端 B 绑定的队列。两个客户端进行点对点的通信时，不是相互之间直接建立连接，而是都通过与服务器建立连接，服务器负责将消息发送给指定客户端。整个过程，消息的传递都是透明的，相互之间不需要知道对方的 IP 地址及端口号。另一种机制是发布/订阅式的消息传递，在发布/订阅模型中，客户端分消息发布者和消息订阅者两种角色。模型定义了如何向一个内容节点发布和订阅消息，这些节点被称作主题（topic）。主题可以被认为是消息的传输中介，发布者（publisher）发布消息到主题，订阅者（subscribe）从主题订阅消息。主题使得消息订阅者和消息发布者保持互相独立，不需要接触即可保证消息的传送。

保证消息可靠性，只需要解决如下 3 个问题：

①消息丢失：接收方未收到发送方发送的消息，而发送方未重复发送。

②消息重复：接收方收到发送方同一个消息的多次拷贝。

③消息乱序：接收方接收消息的次序与发送方发送的顺序不一致。

3.1.3　Web 服务可靠性

目前，OASIS 组织发布了针对 Web 服务技术体系中 SOAP 报文的可靠性规范：WS-Reliability 和 WS-ReliableMessaging 规范。

3.1.3.1 WS-Reliability

WS-Reliability 标准由 SUN、Oracle 等公司发起，是针对 Web 服务可靠性的规范，目前已更新到 1.1 版本，它是以提高互联网上进行的 Web 服务通信可靠性为目的的规范。在交换信息方面，可使用持续性保证、发送保证、避免重复、顺序保护、发送状态确认等功能。

WS-Reliability 标准的目的在于通过定义客户机服务器之间的通信机制，使 Web 服务满足如下要求：

①情况 A（至少一次递交）：发送方发送的消息，接收方至少要递交一次。

②情况 B（至多一次递交）：发送方发送的消息，接收方至多要递交一次。

③情况 C（有且仅有一次递交）：发送方发送的消息，接收方递交且仅递交一次，是情况 A 和 B 的交集。

④情况 D（按照次序递交消息）：在情况 C 的基础上，收到的消息要按照 ID 序号递增的方式，在接收方排序后，再递交给所需交付的应用程序。

WS-Reliability 提出了一种基于消息/确认(message/acknowledgment)模型的可靠性解决方案。这意味着每个消息发送者在发送一条消息后都会等待从消息接收者返回的一条回应消息以确认。通过在 SOAP Envelope 的 SOAP Header 元素中增加一些用于可靠性保证的标签来保证可靠性，如丢弃重复消息的 wsrm：DuplicateElimination 元素、消息过期的 wsrm:ExpiryTime 元素、消息有序的 wsrm:MessageOrder 元素等。图 6 所示是请求消息中的 SOAP Header。

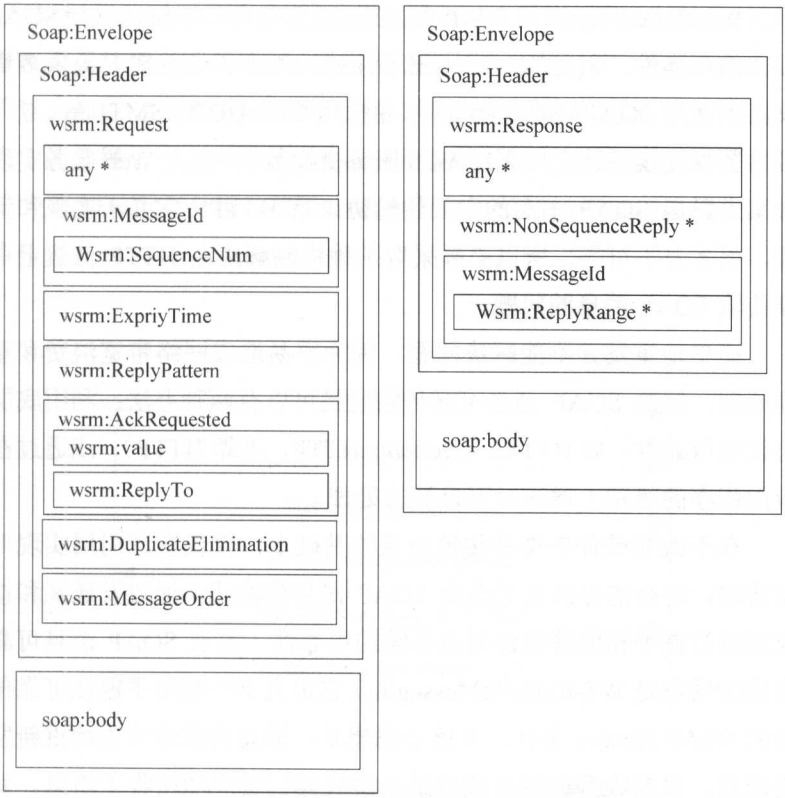

图 6　WS-Reliability 中扩展的 SOAP Header

注：* 该元素可以在报文中出现不止一次。

3.1.3.2　WS-ReliableMessaging

WS-ReliableMessaging 规范是 IBM、Microsoft 等公司发起的同样针对 Web 服务可靠性制定的规范，其功能及实现与 WS-Reliability

标准类似。

WS-Reliability 规范 WS-Reliability 规范基本可以解决采用 SOAP 报文的可靠性，但是实际应用出于性能的考虑，往往需要提供多协议（不使用 SOAP）的直接支持，例如 TCP、UDP、SMTP 等。

数据交换系统之间采用 Web 服务交换数据。而在 Web 服务实现方案中发送 SOAP 消息的主要传输协议是 HTTP，它易于实现和管理，但本身不可靠，所以必须采取某种机制解决在 HTTP 协议上可靠传输 SOAP 消息的问题。

简单地重发并不能解决问题，因为容易造成网络带宽浪费和重复处理。解决 SOAP 消息可靠传输问题可以有两种办法：利用底层协议的可靠性，如 HTTPR（Reliable HTTP，可靠 HTTP），或通过在应用程序的级别上增加对可靠性的处理。

在不改变硬件条件和传输协议的基础上，仅仅在应用层实现可靠传输，可行的办法就是改变 SOAP 消息的格式，通过发送端和接收端进行握手和应答保证消息传输的可靠性。关于 SOAP 消息可靠传输的规范是 WS-ReliableMessaging，它定义了一套用于保证可靠传输的 SOAP Header 条目。其核心思想是：消息的发送方连续重新发送消息，直到收到证实（或确认）接收方已经成功接收了消息。发送方持续发送消息直到接收到该消息的 ACK，接收方不仅要在收到消息后立刻返回 ACK，而且必须忽略任何重复的消息。图 7 说明了 WS-ReliableMessaging 可靠性原理。

端点 A 要向端点 B 发送三个 SOAP 消息分组，A 可以首先连续发送三个独立的分组，在每个 SOAP 消息片断中设置<MessageNum>分别为 1、2、3，且在编号为 3 的 SOAP 消息的 Header 条目中携带</LastMsg>以表明第三个分组是要传输的最后一个分组。假设由于

网络原因第二个分组没有被接收，此时 B 对编号为 1、3 的分组用<AcknowledgementRange>元素进行确认；A 收到确认信息后，发现没有对第二个分组的确认，重新发送编号为 2 的分组，同时在 SOAP 消息头部用<AckRequested>元素要求确认；B 接收到 A 重发的编号为 2 的分组用<Acknowledgement>元素对 1、2、3 进行确认；A 收到 B 发来的对所有分组的确认，认为所有已发分组都被正确接收，结束传输。

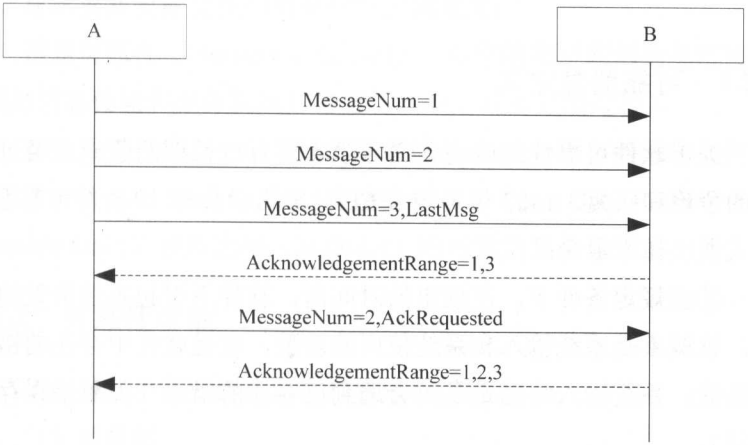

图 7　WS-ReliableMessaging 可靠性原理

TCP 协议通常采取消息确认、超时重传、流量控制和拥塞控制等措施来处理消息的可靠性问题；JMS 作为目前应用比较广泛的一种消息服务，它定义了消息传输的规则，并定义了与应用无关的消息服务接口，在消息传输的可靠性方面，它主要关注消息丢失、消息重复以及消息乱序几个方面；在 Web 服务可靠性上，WS-Reliability 和 WS-ReliableMessaging 标准主要应用于 SOAP 消息层，通过在

SOAP 信封的 SOAP Header 元素中增加保障可靠性的标签，有效地解决了 SOAP 协议层上关于可靠性的问题。数据交换主要关注基于 Web 服务技术如何在发送方、中间环节、接收方之间实现数据通信，因此需要分析数据交换可靠性的参数，研究影响数据交换可靠性的因素，并提出一套可靠的消息机制来保证数据交换的可靠传输。

3.2 数据交换的可靠性

3.2.1 可靠消息定义

关于软件可靠性的确切含义，学术界有过长期的争论，经过长期的争论和研究，1983 年美国 IEEE 计算机学会对 “软件可靠性” 正式做出如下定义：

①在规定条件下，在规定的时间内，软件不引起系统失效的概率，该概率是系统输入和系统使用的函数，也是软件中存在的错误的函数；系统输入将确定是否会遇到已存在的错误（如果错误存在的话）。

②在规定的时间周期内，在所述条件下程序执行所要求的功能的能力。

这一“可靠性”的概念是对于计算机系统中的软件和硬件的整体来定义的。至今，针对可靠性的研究已近半个世纪，可靠性是一个综合特性，其研究涵盖硬件、网络、操作系统、数据库、应用系统等各个层次。在特定的环境中，可靠性拥有其具体的含义。

在传统的分布式系统中，Andrew S. Tanenbaum 将可靠性刻画为可用性、安全性和容错性这三个特性的集合体。其中，①可用性是

指系统功能可以满足用户需求，且能稳定运行不出错误；②安全性是指系统允许合法用户在其授权范围内对系统资源进行访问和利用，且系统应能抵制和防止各种非法用户的侵入；③容错性是指系统在出现故障的情况下仍能持续工作的能力。

由于基于 Web 服务技术的数据交换网络主要应用于分布式环境中，因而针对基于 Web 服务的数据交换消息可靠性的定义也源于传统分布式系统对可靠性的定义；但与后者相比，有了更为具体的内涵。在这里从数据交换的消息层给出其定义：

消息可靠性（Message Reliability）：可靠的和可预测的基础服务（例如消息传输和服务发现）的交付（双方或多方的过程）。

在数据交换中，主要关注的是如何基于 Web 服务的消息如何满足至多一次（At Most Once）、至少一次（At Least Once）、恰好一次（Exactly Once）和按次序（In Order）的可靠消息传递需求。

3.2.2　可靠性参数

下面对几个主要的可靠性参数进行讨论。

（1）可靠度

可靠度 R 是指数据交换中心在规定的条件下、规定的时间段内完成预定的消息传输功能的概率，或者说是数据交换中心在规定时间内无失效发生的概率。

用随机变数 ξ 表示从数据交换中心运行开始到系统失效所经历的时间，用 $F_\xi(t)$ 表示 ξ 的分布函数，用 t 表示任意给定的时刻，用 $R_\xi(t)$ 表示数据交换中心在 t 时刻的可靠度，则数学公式如下：

$$R_\xi(t) = P_r\{\xi > t\} = 1 - F_\xi(t) \tag{3.1}$$

（2）失效率

失效率是指数据交换中心在 t 时刻没有发生失效的条件下，在 $(t+\Delta t)$ 区间内，当 Δt 非常小时，单位时间内发生失效的概率。

用随机变数 ξ 表示从数据交换中心运行开始到系统失效所经历的时间，用 t 表示任意给定的时刻，用 $\lambda(t)$ 表示失效率，则失效率数学公式为：

$$\lambda(t)=\lim_{\Delta t\to 0}\frac{P_r\{t+\Delta t\geqslant\xi>t/\xi>t\}}{\Delta t} \tag{3.2}$$

（3）平均失效间隔时间（MTBF）

MTBF 是指两次相邻失效时间间隔的均值。假设当两次相邻失效时间间隔为 ξ，ξ 具有累计概率密度函数 $F(t)=P(\xi\leqslant t)$，即可靠度函数

$$R(t)=1-F(t)=P(\xi>t)$$

则

$$\mathrm{MTTF}=\int_0^{\infty}R(t)\mathrm{d}t \tag{3.3}$$

3.2.3 可靠性因素分析

基于对网络状况、操作系统故障、硬件设施等因素的考虑，需要为数据交换系统设计一套可靠消息机制，以保证能够可靠地完成数据交换。

数据交换在应用中具有不可抗拒的外在因素作用，如系统（系统当机、进程被终止、临时断电）、网络（连接、网络状况）、硬件因素（硬盘、CPU、内存）等。其中操作系统当机、进程被终止、机器断电等是任何系统都不可避免的；网络故障也是分布式应用中不可避免的情况，包括无法连接到网络、通信目标无法连接、网络

状态差等；由于系统的处理逻辑异常和负载能力会引起如处理流程异常、内存溢出等。

为了消除网络、系统、硬件等因素所造成的潜在危险，要求数据交换系统必须具有可靠性保障。要解决数据交换的可靠性问题，首先需要明确定位哪些因素导致数据交换的不可靠。从数据交换平台的部署结构示意图（见图 8）可以看出，这些不可靠性因素主要位于如下 4 个位置：

①上层应用与数据交换平台的接入端：当上层应用程序与平台接入时，避免不了会出现无效错误的请求，如无法找到资源、无法定位目标、格式不正确等。

②数据交换平台所处的运行环境：平台本身处于的运行条件包括所选择的操作系统、网络连接状况、CPU 状况、内存空间、磁盘空间等因素，这些因素都有可能导致系统不能正常运行。

③数据交换平台本身：数据交换平台本身是否具备一定的容错和故障恢复能力，是否能够进行负载均衡等。

④平台间传输的消息：平台间所传输的业务信息是否出现丢失、延迟、重复以及乱序等情况。

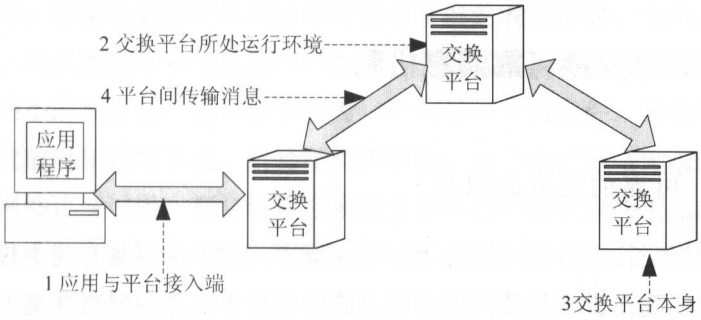

图 8　数据交换平台的部署结构示意

通过前面的分析可以看出，数据交换中这些不可靠的因素主要集中在系统运行支撑、消息协议两个方面（见图9）。其中系统运行支撑既包括数据交换平台可靠性的支持，也包含对上层应用接入的可靠性的支持；消息协议是指对网络上传递的消息的处理。而由于系统运行支撑和具体的系统相关，这里不予以讨论，本书主要关注消息协议。

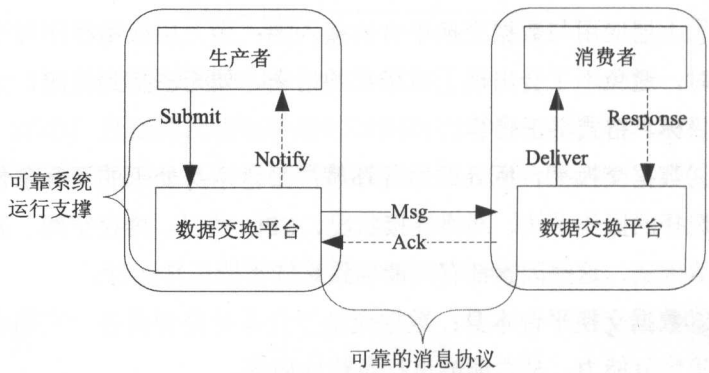

图9　数据交换平台可靠性体现示意

3.3　数据交换可靠消息机制

3.3.1　可靠消息机制目标

数据交换平台系统运行于网络环境中，由于网络具有的不稳定性，数据交换平台系统需要采取措施保证在不可靠的网络环境下能够将数据安全可靠地在数据交换双方之间传输。根据对消息传输的

分析，可靠消息机制要达到如下目标：

①至少一次消息成功送达的语义。

②最多一次消息成功送达的语义。

③一次并且仅一次消息成功送达的语义。

④一组消息必须按照一定的顺序成功送达的语义。

与网络传输协议相似，数据交换平台系统的可靠消息机制也需要做到消息传送可靠、按序、无丢失、无重复。可以通过如下方式实现：

①可靠传输通过消息序号和消息确认来保证。

②按序可通过消息序号以及数据分片序号来保证。

③无丢失可通过消息重发及消息确认来保证。

④无重复可通过丢弃无效及重复消息来保证。

3.3.2　数据交换可靠消息机制的设计

根据前面的分析，本书中数据交换系统可靠性的设计将主要针对设计可靠消息机制展开。

可靠消息机制是指对网络上数据交换双方传递的消息的处理。其中，消息是数据交换系统进行数据交换的表现形式，它由消息内容、消息发送者和消息接收者组成。在形式上，消息可以表现为一个很小的数据包，也可以表现为一个容量很大的文件。无论是由于系统故障如系统当机、机器断电等事件，还是由于网络无法连接或者网络速度过慢，都会造成在数据交换平台中所交换的消息出现延时、丢失及重复等，这都需要数据交换系统提供对消息的可靠处理，也就是数据交换可靠消息机制主要解决的问题是因为系统或者网络原因导致的消息延迟、丢失、重复等问题。因此，可靠消息机制设

计如下：

①通过定义消息确认可以保障消息没有丢失。

②通过定义在长时间未收到等待的消息时的处理，来进行消息重发。

③由于重发的缘故，必然会出现重复的消息，这些消息会对正常的业务处理逻辑产生干扰，需要识别这些重复的消息并丢弃。

3.3.2.1 消息确认

在数据交换中，为了支持可靠性消息，须确保消息送到了目的地，而这是由消息确认来实现的。也就是说，对接收到的每个请求消息，必然要返回一个响应消息。这个响应消息可以是正常的确认消息，也可以是错误消息。消息发送操作包括推数据操作和拉数据操作两种。在每个操作中，响应消息携带对请求操作的确认信息，从而确认请求消息。在推、拉数据的操作中，响应消息在实现对请求消息有效确认的同时，可以进行一些协商，比如是否要求安全特征、是否接受对方的请求、是否协商数据分片大小等。

因此，在数据交换的过程中，发送方每发送一个请求消息报文，接收方都要向发送方发送相应的响应消息报文，用于确认请求消息，同时附带一些交互性信息。消息确认机制保证了数据消息没有丢失，并可以在发送方和接收方之间传递交互信息。

下面分别介绍推、拉数据操作的过程以及在该过程中所涉及的报文结构。

推数据操作是指一个数据从发送方推送给接收方的过程。推数据操作示意如图 10 所示。

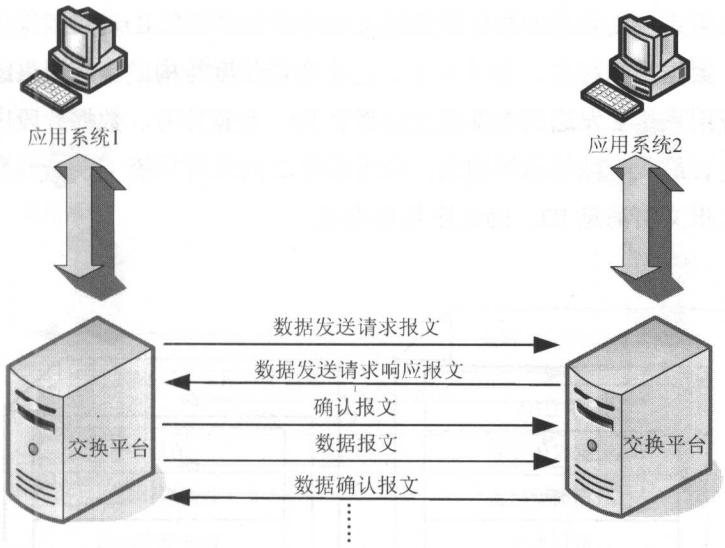

图 10　推数据操作

推数据操作过程描述如下：

①应用系统 1 向应用系统 2 发送数据发送请求报文。

②应用系统 2 在接收到该请求报文后，向应用系统 1 发送数据发送请求响应报文用于进行消息确认。

③应用系统 1 在接收到该响应报文后，才向应用系统 2 发送数据报文。

④应用系统 2 在接收到数据报文后，向应用系统 1 发送数据确认报文从而进行消息确认。

当应用系统 1 向应用系统 2 发送数据时，报文结构如图 11 所示。应用系统 1 向应用系统 2 发送的数据发送请求报文由消息 ID、数据媒体类型、数据类型标志、数据大小、传输模式等构成；应用系统 2

向应用系统1发送的数据发送请求响应报文由消息ID、数据媒体类型、数据类型标志、分片大小、已收数据长度等构成；应用系统1向应用系统2发送的数据报文由消息ID、数据序号、数据片段序号和是否最后一帧标志等构成；应用系统2向应用系统1发送的数据确认报文由消息ID、确认序号等构成。

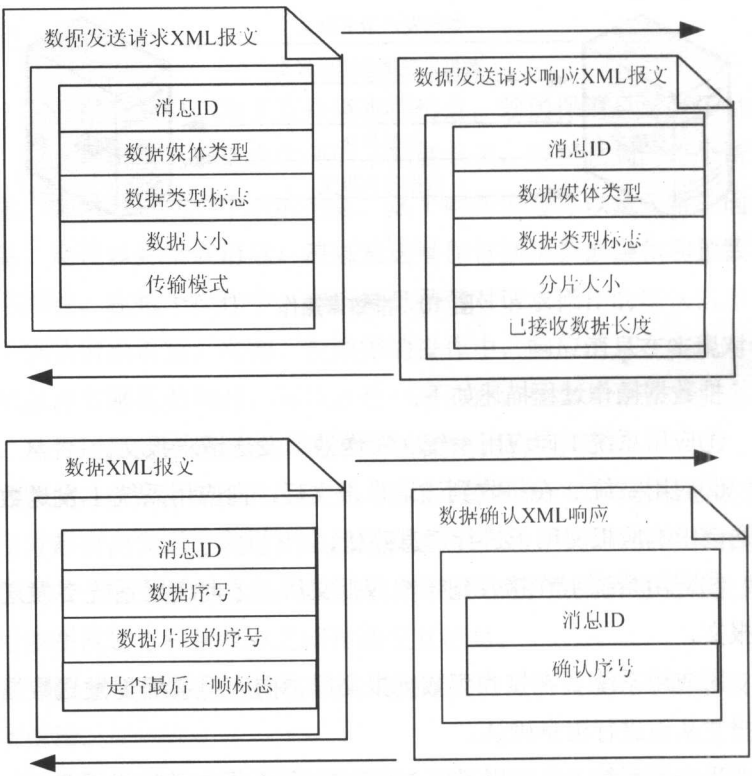

图11 推数据操作报文

拉数据操作是指数据接收方从数据拥有方获取数据的过程。拉数据操作示意如图 12 所示。

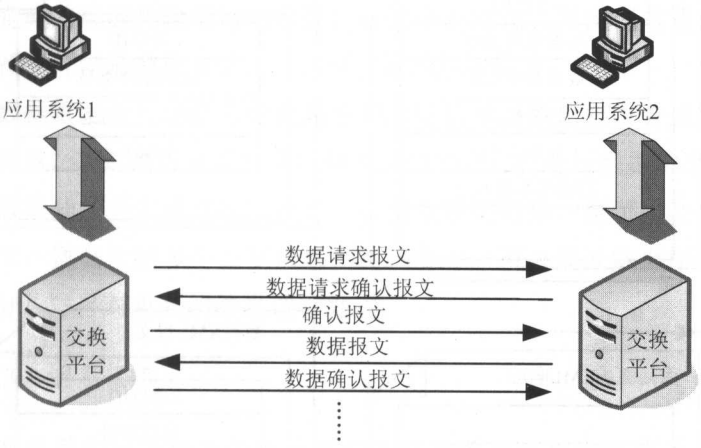

图 12　拉数据操作

拉数据操作过程描述如下：

①应用系统 1 向应用系统 2 发送数据请求报文。

②应用系统 2 在接收到该请求报文后，向应用系统 1 发送数据请求确认报文。

③应用系统 2 向应用系统 1 发送数据报文。

④应用系统 1 在接收到应用系统 2 发送的数据报文后，向应用系统 2 发送数据确认报文，进行消息确认。

拉数据操作报文结构见图 13。

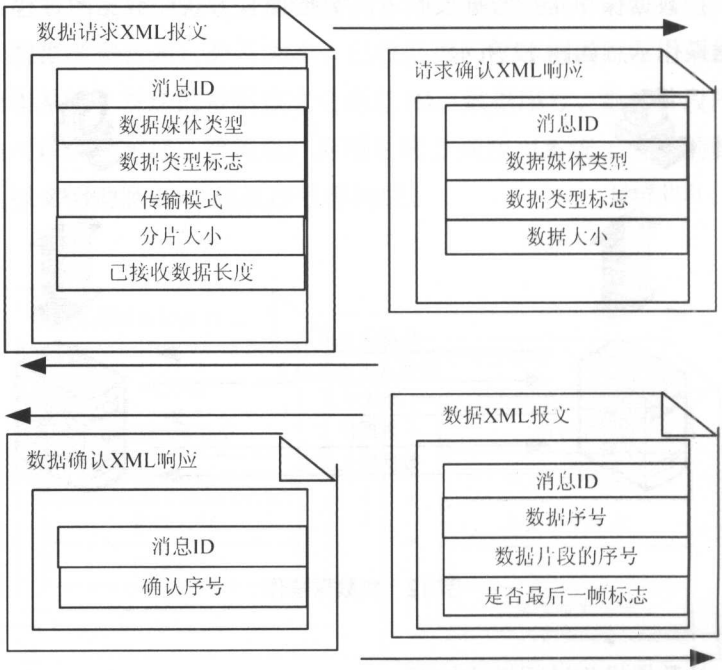

图 13　拉数据操作报文

3.3.2.2　消息重发

数据交换系统每发送一条消息，就设置一次计时器并对该条消息进行计时。如果在计时器设置的挂起时间内没有接收到确认消息，就挂起该条消息所对应的会话，并结合当时系统及网络状况，进行消息重发。

消息重发分为请求重发和数据重发。在数据请求操作中，消息重发是指重发请求消息；在数据传输操作中，消息重发是指重发数据。

请求重发是指发送方因系统或网络的原因而长时间没有接收到接收方的响应，重新发送请求消息，或者接收方在系统空闲时为恢复未完成的数据传输而发出继续传输的请求消息。因此消息重发的执行者既可以是消息发送方，也可以是消息接收方。

数据重发是指发送方根据系统和网络状况以及由接收方返回的确认消息，决定是否重新发送已经发送的消息。在进行数据传递时，如果数据分片丢失或延时，发送方会再次发送数据，这可能会导致包含同一数据片的多个重复消息到达接收方。可以通过数据序号和数据分片序号对重复的消息进行处理。

3.3.2.3　消息丢弃

消息丢弃可以分为无效消息的丢弃和重复消息的丢弃两种情况。

针对无效消息的丢弃，如果接收到重复的消息，丢弃重复的消息，并只确认有效的消息。会话在整个数据交换过程中具有不同的状态，可以根据该会话所处的状态来确定是否接收消息以及消息的类型，从而可以识别出接收的消息是否重复以及是否有效。例如，处于读取数据的状态的时候却收到了一个消息报文，此时由于该报文处于非正常接收状态因此不会被处理。

数据交换系统中还定义了可靠消息机制的时间特征。可靠消息机制的时间特征是指消息从发送方发送到接收方，接收方接收到消息后返回给发送方相应的确认消息的整个时间间隔。超过时间间隔的消息被视为无效消息，应该被接收者丢弃。传输的消息报文中具有过期时间标记，在消息过期后，此消息将会被丢弃。

对于重复消息的丢弃，由于在传输数据的操作中，根据数据序

号和数据分片序号可以有效识别接收的重复数据，通过比较可丢弃重复的消息。

3.3.3 数据交换可靠消息机制的实现

本节讨论了可靠消息机制实现的一些技术，它主要体现在状态协议管理和可靠消息报文两部分。协议状态管理包含对消息重发、消息丢弃的处理逻辑，主要由可靠消息处理机来完成，消息报文是消息确认、消息重发及消息丢弃的基础。

3.3.3.1 可靠消息处理机制

数据交换的动作可以划分为请求数据操作、数据通知操作、推数据操作三种。其中，请求数据操作和推数据操作的组合可以实现实体 A 到实体 B 获取数据；数据通知操作和推数据操作的组合可以实现实体 A 把数据发送到实体 B。每种组合都有一个数据生产者和一个数据消费者。数据生产者是指拥有数据的节点，数据消费者是指需求数据的节点，其分别拥有自己的处理逻辑。因此两种组合共存在四套处理逻辑。

状态机指的是一次数据交换的数据消费者和数据生产者的处理逻辑，它由一系列的状态组成，而每个状态具有一定的系统特征，并且具有一定的处理能力。状态机在收到消息报文后首先进行一定的处理，通过判断消息是否有效去除无效的消息，通过判断消息是否超时来去除延时的消息，然后再进行具体的业务逻辑处理。图 14 描述了状态机在接收到消息之后的处理流程。

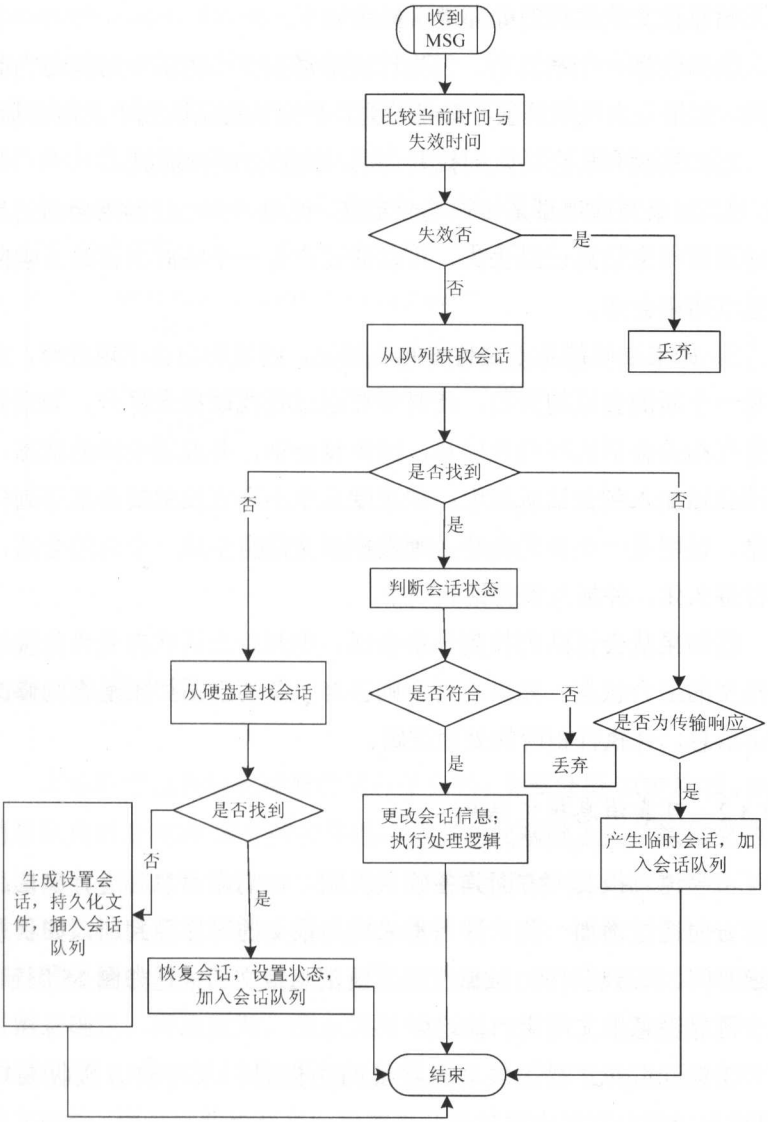

图 14　接收消息报文处理流程

消息报文到达后的处理逻辑描述如下：

①当收到一个消息后，首先比较消息报文中的失效时间与当前时间，如果报文失效则丢弃该条消息，否则从会话队列中获取会话。

②如果没有从会话队列找到会话，则要分两种情况：

A. 如果当前消息是传输响应消息，说明一个已经结束会话的最后的通信结束信息已经丢失，可以通过产生一个临时会话给出响应消息以结束会话。

B. 如果当前消息不是传输响应消息，则说明会话出现故障，或者是一个新的会话的开始，此时需要通过查找硬盘来区分。如果硬盘存在相关会话的序列化信息，则恢复会话，并设置会话的状态，把该会话加入到会话队列中；如果硬盘中不存在相关的会话序列化信息，说明是一个新的会话，则根据报文信息生成一个新的会话，进行持久化，并加入到会话队列中。

③如果从会话队列找到该条会话，则判断会话状态是否是接收该报文的适合状态，如果不适合则丢弃该条消息；如果适合则修改会话信息，并执行相应的处理逻辑。

3.3.3.2 可靠消息报文结构

可靠消息报文一方面具备确认机制，在响应消息时进行确认；另一方面通过增加一些关键元素来提高报文的可靠性判断，如状态持续时间、消息超时等。数据交换系统消息报文的描述如图 15 所示。

可靠消息报文结构的描述如下：

①Routing 元素提供消息寻址路由信息以及接收方接收端口信息。

② PullRequest、PullResponse、PushRequest、PushResponse、

TransportRequest、TransportResponse、IIPFault 等元素每次只能出现一个，分别表示数据交换系统交换的报文。在数据交换系统中，定义了三种操作：请求数据操作、数据通知操作、推数据操作，而每个基本操作都由请求消息和响应消息组成。此外，定义了错误信息报文。

③Features 元素描述了系统的一些特征信息，比如安全信息等。

```
<IIPMessage SessionID="" SessionType="">
    <Routing/>
    <PullRequest/>|<PullResponse/>|<PushRequest>|<PushResponse/>|
<TransportRequest>|<TransportResponse>|<IIPFault/>
    <Features/>
</IIPMessage>
```

图 15　可靠消息报文结构

3.4　本章小结

本章对消息传输的可靠性进行了分析，主要分析 TCP 协议、JMS 消息队列以及 Web 服务的可靠性，然后在此基础上，从数据交换消息层定义了消息可靠性，讨论了消息可靠性的参数，并对基于 Web 服务的数据交换框架的不可靠因素进行了分析，指出了消息传输可靠性需要解决的问题，提出了可靠的消息机制的设计目标，主要针对消息确认、消息重发、消息丢弃等方面，设计了可靠的数据交换消息机制，并讨论了其实现的一些技术，如可靠消息处理机及可靠报文结构，通过这些措施来保障数据交换过程中数据传输的可靠性，满足至多一次、至少一次、恰好一次和按次序的可靠消息传递需求。

第 4 章
数据交换效率研究

数据交换可能发生在任何两个应用系统之间，交换双方可能处于同一个局域网内，也可能处于广域网上相距甚远的两个网络节点上，交换过程可能跨越多个网络自治域，不同的应用及不同的网络都会对数据交换的效率产生一些影响。数据交换基于 Web 服务，Web 服务本身也会对数据交换的效率产生影响。本章首先对 Web 服务传输性能进行分析，分析其影响 Web 服务性能的因素，并提出了相应的优化方案，设计了类型映射机制和基于滑动窗口的 SOAP 消息传输机制来用于改进基于 Web 服务的数据交换的效率。

4.1 Web 服务传输性能分析

随着互联网软件技术及其应用的迅速发展，基于 Web 服务的分布式计算模式日益成为软件技术和应用发展的趋势，Web 服务为分布式计算提供了一种新的范例。目前许多基于 Web 的分布式应用（如电子商务、电子政务等）已将 Web 服务作为其架构的关键技术基础。

Web 服务是基于 Web 的应用模式，使用开放的、基于 XML 的标准和传输协议与客户端交换数据。Web 服务正成为不同企业的分

布式应用之间通信的最重要的技术。即使在一个企业内部，由于 XML 得到了更广泛的应用，Web 服务也正成为事实上的技术标准。由于 Web 服务正成为主流的技术，对于用户来说了解 Web 服务的特征诸如性能等是非常重要的一件事情。

Web 服务是一种可使不同用户通过互联网实现共享数据和应用功能的手段，它使得 Internet 范围内的应用系统能够更有效地实现资源共享、信息交换和协作。Web 服务技术建立在 XML 技术基础之上，包括由于 XML 本身的特点（与二进制表示相比具有较大的报文长度、较长的解析时间）及其他一些限制因素，造成了 Web 服务技术与其他分布式计算技术（如 DCOM、CORBA）相比性能具有一定的差距。而电子商务由于其交易的实时性等特点，要求基于 Web 服务的电子商务系统具有较高的性能，这样性能就成为决定 Web 服务能否得到进一步广泛应用的关键因素之一。目前，如何提高 Web 服务传输的性能正成为 Web 服务技术的一个研究热点。

关于 Web 服务传输性能的研究，近年来在学术界和工业界已经取得了一些成果。有些研究者提出了在客户端缓存 SOAP 消息的机制来优化 Web 性能。有些研究者使用二进制表示来包装 SOAP 报文。有些研究者通过压缩 SOAP 来改善 Web 服务的性能。这些相关研究的成果在一定程度上改善了 Web 服务性能，但同时也带来一些相应的负面影响。例如，使用二进制格式来对 SOAP 报文进行编码，丧失了基于 XML 的 SOAP 报文的原有的优势如可读性；压缩 SOAP 报文会造成新的压缩/解压时间的开支，并且使应用的访问变得不透明。

本节分析了 Web 服务性能的现状，结合 Web 服务调用模式综合分析了影响 Web 服务性能的因素，在此基础之上，从提高 Web 服务

整体性能出发，提出了一些优化 Web 服务性能的策略，把 Web 服务性能优化分为传输协议层、SOAP 实现层、Web 服务运行环境三个层次，并针对每个层次分析了 Web 服务性能优化的核心支撑技术。

4.1.1 Web 服务性能现状分析

Web 服务是在 Internet 上通过标准协议访问服务的一种计算模式，它是面向服务的体系结构的一种实现方式。Web 服务与其他分布式技术相比的优势在于它的互操作性、普遍性和行业支持。同时意味着，在性能衡量指标一样的情况下，Web 服务性能与其他分布式技术相比也具有一定的差异。本小节首先介绍 Web 服务应用模式，然后介绍 Web 服务性能的衡量指标，最后利用这些衡量指标比较 Web 服务与其他分布式技术的性能。

4.1.1.1 Web 服务性能的衡量指标

性能是指计算机系统或子系统实现其功能的能力以及对计算机系统或子系统执行其功能的能力的度量。例如，响应时间、事务处理能力、可靠性等。性能通过吞吐率和延迟来衡量。高的吞吐率和低的响应时间表示良好的性能。吞吐率表示在一个给定时间内处理的服务数量；反应时间表示从发送请求到接收响应所需的时间。

Web 服务性能关注的是在 Internet 环境中 Web 服务运行环境实现 Web 服务调用功能的能力以及其执行功能的能力的度量。Web 服务性能可用各种方法来度量，但最终归结为用户请求到响应所需要的时间。Web 服务性能的基本计量单位是从一个用户请求到响应所需要的时间，目的就是在一个很短的确定性的时间段内实现请求。Web 服务性能使用延迟和吞吐率来衡量。

Web 服务使用 SOAP 协议进行服务调用，而 Web 服务请求的 SOAP 封装、报文解析、编码方式、解析方式、Web 服务粒度等会对 Web 服务性能造成一定的影响。因此需要对 Web 服务性能进行分析。Web 服务性能分析就是分析 Web 服务底层协议对 Web 服务性能的影响，分析在执行某一确定的协议时，网络所表现出来的性能情况。Web 服务性能分析研究的目的在于找出影响 Web 服务性能的因素，从而针对问题采取进一步改进协议及提高性能的措施。

4.1.1.2　Web 服务与其他分布式技术性能的比较

尽管 Web 服务技术已经逐步成熟，并且也得到了广泛的支持，但是目前 Web 服务技术的实际应用却不让人感到乐观，这是由于 Web 服务体系结构仍存在着一些局限性，如安全性、管理、消息处理、性能等。2002 年 7 月 North American Operations 称性能是 Web 服务体系结构现有局限性问题之一。

Web 服务的延迟及吞吐率与其他分布式技术（如 CORBA、JAVA RMI、DCOM）相比具有一定的差距。有些研究者对 SOAP 的实现与 CORBA、DCOM、JAVA RMI 技术的实现的延迟及吞吐率分别进行了比较。图 16 是客户端发起一个简单的调用，这些实现之间的延迟时间的比较。从中可以看出 SOAP 的实现，无论是 Apache SOAP 抑或最新版本的 Axis，它们的延迟时间都要与其他几种分布式技术的延迟时间要长。

Web 服务性能之所以不容乐观，是由于 Web 服务技术的特点（如使用 XML 作为数据编码基础、主要与 HTTP 协议绑定等），处理服务请求需要耗费更多的服务器系统资源，造成 Web 服务调用存在较严重的性能瓶颈，无法满足实际应用的需要，从而使 Web 服务性能

成为目前 Web 服务尚未能够得到广泛应用的原因之一。因此研究影响 Web 服务性能的因素，使用相关技术优化 Web 服务性能具有必要性和急迫性。

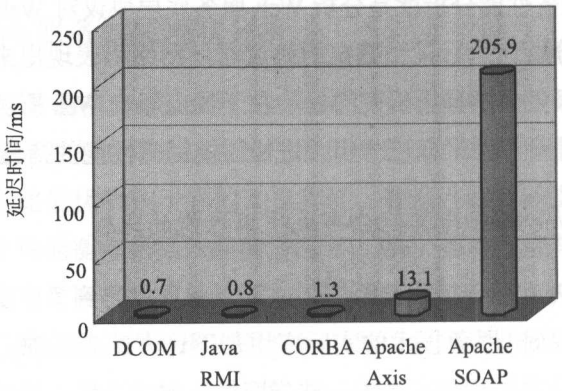

图 16　几种分布式技术延迟时间的比较

4.1.2　影响 Web 服务性能的因素

随着 Web 服务的应用日益广泛，用户在享受 Web 服务所带来的互操作性、松散耦合性、可集成性等好处的同时，也面临着由于 Web 服务性能低下所带来的负面影响。Web 服务的两大特性是：基于标准协议（XML、SOAP 等）和分布式。基于标准协议如 XML，会涉及 XML 报文的解析、绑定、验证、转换等操作；基于分布式，会涉及网络带宽、路由、客户端处理、服务器端的处理等方面。而这些方面都可能会对 Web 服务的性能造成影响。下面在分析 Web 服务调用模型的基础之上，从服务传输层、SOAP 协议实现、Web 服务运行环境三个层次对影响 Web 服务性能的因素进行了分析。

4.1.2.1　Web 服务调用模型

Web 服务调用是 Web 服务客户端通过网络调用服务器端所部署的 Web 服务的过程。Web 服务调用模型如图 17 所示。Web 服务客户端在调用一个 Web 服务的时候，需要构建并发送请求 SOAP 报文；服务器端在接收到请求报文之后，要对报文进行解析、验证，调用 Web 服务的实现，然后构建并发送响应 SOAP 报文；Web 服务客户端在接收到响应 SOAP 报文之后，同样需要对报文解析、验证，供应用程序进一步处理。

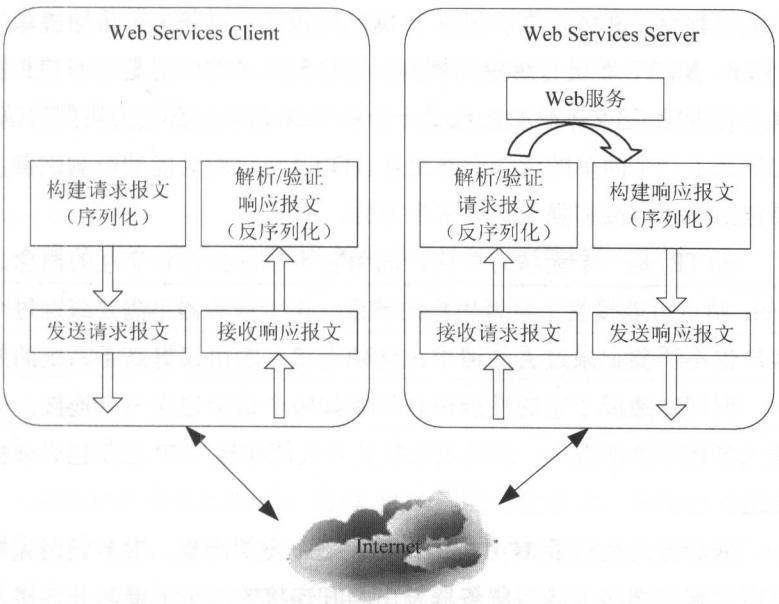

图 17　Web 服务调用模型

通过分析 Web 服务调用模型，可以看出：Web 服务调用过程主要包括报文的传输、SOAP 报文的处理（包括解析、验证、转换等）、Web 服务运行环境对 Web 服务处理的支持等几个部分。因此，为了清晰和方便起见，本书把对 Web 服务性能的影响因素分为三个层次：服务传输层、SOAP 协议实现层和 Web 服务运行环境层，下面将从这三个层次来分析影响 Web 服务性能的因素。

4.1.2.2 服务传输层

在 Web 服务调用过程中，传输协议、网络带宽等因素会对服务传输的性能造成重要影响。在这里，主要分析传输协议对 Web 服务性能的影响。SOAP 协议制定了 SOAP 报文可以与多种传输协议如 HTTP、SMTP 等进行绑定的规则，但目前更多的应用是与 HTTP 协议进行绑定。超文本传输协议是一个在 TCP 协议顶部的应用层协议，它定义了一个简单的请求应答交互。HTTP 协议对性能的影响因素主要包括永久连接和强制延迟两个方面。

HTTP 是一个无状态协议，因为它不包含会话和交互的概念。Web 站点可能接收上百万用户的请求，由于服务器不需要跟踪每个客户也不需要记录过去的请求，这种无状态的协议提高了系统的效率，但同时造成了它的性能消耗。例如每个请求建立一个连接，即使大多数对象都很小，但是用来打开和关闭连接 TCP 控制包必须被发送。

强制延迟是衡量 HTTP 协议的另一个主要因素。服务调用完毕之后，客户端必须等待服务器发出断开连接的请求才能断开连接，造成了一定的连接和请求延迟。

4.1.2.3　SOAP 协议实现

SOAP 是一种在分布式环境下用来交换结构化信息的轻量级协议，它的实现包括 SOAP 报文的解析、验证、类型映射以及 Web 服务安全性处理等几个方面。目前在基于 Web 服务的解决方案中，SOAP 协议实现对 Web 服务性能的影响主要表现在以下几个方面：

①SOAP 报文的解析：SOAP 协议建立在 XML 技术的基础之上，要求编码格式是 ASCII 文本。这是使用 SOAP 的最大的优势，因为应用程序在通信之前不需要彼此协商。然而，既然编码格式是 ASCII 文本，所有的 SOAP 消息中的自描述信息都需要被转换为 ASCII 字符串的形式，传输的时候要从 ASCII 格式转换为二进制编码，会花费大量的转换的代价，并且会造成网络传输的高成本，因为 ASCII 编码的消息比二进制的原始消息要大（见表 1），解析它所需要花费的时间就会更长。

表 1　二进制和 XML 表示报文长度的比较

Java 变量	short year=2004;	长度
二进制表示	0x07DE	2 字节
XML 表示	<year>2004</year>	17 字节

②SOAP 报文的验证：通过 SOAP 报文的验证，来决定 SOAP 报文是否有效，也就是决定报文是否遵循所制定的 schema。SOAP 报文验证包括 schema 文档的读取、解析以及比较。schema 文档通常是基于 XML 协议的，而 XML 文档的处理原本就比较耗时，尤其在 schema 文档不在本地的情况下，会更大程度地增加时间开支。

③类型映射：类型映射涉及目标系统的数据类型到XML类型的序列化和XML类型到目标系统的数据类型的反序列化。SOAP消息的结构越复杂，涉及的类型越多，程序设计的对象和XML元素之间的映射所需要的时间就越长。

④Web服务安全性处理：为了保证应用程序终端之间的安全性，Web服务会增加一些安全处理特性，这些安全处理特性包括访问控制、Web服务加密、Web服务数字签名、Web服务授权等功能。这些安全处理特性无一例外都需要对SOAP报文功能处理。而这些安全特性可能会惊人地延长处理服务请求的时间，降低Web服务的性能。

4.1.2.4 Web服务运行环境

Web服务运行环境提供Web服务的部署、运行、调用、管理等功能，它的体系结构的设计和实现的好坏会对Web服务性能产生重要影响。其中，能否把请求按照一定的规则分配到多个服务器，能否根据用户的需求为用户提供相应级别的服务，是Web服务性能是否能够进一步提升的重要因素。

4.1.3 Web服务性能优化方案

根据上述对影响Web服务性能的因素的分析，本研究提出了Web服务性能优化方案。在该方案中，阐述了Web服务性能优化的策略，并着重分析了性能优化的措施和技术以及它们的优劣。

4.1.3.1 Web服务性能优化的策略

Web服务性能优化的目的是找出并解决影响Web服务性能的瓶

颈，提升 Web 服务性能，保障能够正常调用服务。在 Web 服务性能优化研究的过程中，本研究提出了以下优化策略：

①保持原有优势：Web 服务的最大优势在于互操作性。在性能优化的过程中，不能为了优化性能而使用一些专用的协议从而失去 Web 服务的优势，这样就失去了优化的意义。

②设计优先介入：设计阶段优先考虑 Web 服务性能，会比以后再作优化要更有效。比如在 Web 服务接口设计的时候，要考虑服务的粒度。应该发布本身能够接受的所有必需的参数和信息的粗粒度服务，从而使得服务提供者能够提供尽可能多的服务，目的是将完成一组业务任务所发出的请求数目减到最少，这样避免了不必要的消息交换，从而将对 Web 服务性能的影响降到最小。

③整体分析，区别对待：Web 服务性能的优化要从整体上考虑，否则尽管采取某种优化措施可能会取得一定的成效，但是可能会造成其他的一些负面影响；此外针对具体情况，也可以采取一些特殊的处理方法，比如在双方协商的基础之上压缩报文从而减轻网络传输负荷。

在遵循上述优化策略的前提之下，针对 Web 服务性能影响因素，分服务传输、SOAP 协议实现、Web 服务运行环境三个层次提出了性能优化的措施和技术，如表 2 所示。

4.1.3.2　选择合适的传输协议

HTTP 的永久连接特性是一种要求 HTTP 连接持续的方法。永久连接对于大量的小报文而言是一种较好的选择，对于大报文，影响不大。在大报文传输的情况下，流连接可能会具有较好的性能，而 HTTP 的块编码本身就是一种流。

表2 Web服务性能优化的措施和技术

层次	影响因素	措施和技术
选择合适的传输协议	无会话	http/1.1协议；永久连接；块编码
	强制延迟	客户端主动关闭连接
优化SOAP协议实现	SOAP报文解析	更好的XML解析技术（二进制编码、压缩、缓存）
	SOAP报文验证	减少验证、本地验证
	类型映射	采用document/Literal类型编码方式
	Web服务安全	传输层的安全机制
改进Web服务运行环境	服务过载	负载均衡
	区别服务	服务分级

服务器端的强制延迟是指只有服务器提出关闭连接的请求，客户端才会关闭连接，否则客户端一直会处于等待状态。将客户端置于主动关闭状态，即客户端可以单方关闭连接，可以减少延迟。

HTTP1.1缺省支持永久连接和块编码，因此选择HTTP1.1并且设置客户端主动关闭状态，会比选择HTTP1.0的传输效率更高。

4.1.3.3 优化SOAP协议实现

SOAP协议是Web服务的消息层协议，对SOAP协议实现的优化尤显重要。SOAP协议实现的优化主要是从对SOAP报文的操作入手，主要包括XML解析、XML验证、类型映射和Web服务安全等。

SOAP报文的解析需要使用更好的解析技术。对于多数应用而言，事件驱动的解析器SAX（Simple API for XML，用于处理XML事件驱动的推模型）比DOM类型的解析器具有更好的性能。大的SOAP消息是一个性能瓶颈，因为需要花费大量的时间来解析它们。

因此使负载的尺寸尽可能小。对报文的内容进行二进制编码、压缩报文、在客户端缓存 SOAP 请求报文或者在服务器端缓存响应报文，能够减少请求报文长度或者报文数量，有助于提高性能。但它们也存在着一些弊端，如二进制编码需要通信双方协商编码格式，压缩需要通信双方协商压缩算法并且会增加报文解析时间，缓存需要考虑保证数据是否是最新的。

对于验证过的 SOAP 报文进行标记，下次再遇到类似的报文不再进行验证。报文即使要进行验证，也要尽量保证 schema 文档在本地，进行本地验证，减少通过网络验证的传输开销。

Web 服务的类型映射包括 Document、Literal、RPC 三种编码方式。使用 Document 和 Literal 编码方式，负载的复杂度低，减少了序列化/反序列化报文的时间，因此优先选用 Document/Literal 编码方式。

Web 服务安全需要更大的开支。并非所有的 SOAP 通信都需要采取安全措施，在同一个域中通信就可以降低安全级别。在多数情况下，端到端的安全（如 WS-Security）的性能开支要比传输层的安全机制如 SSL 要高，这个时候可以考虑采用 SSL。

4.1.3.4　改进 Web 服务运行环境

Web 服务运行环境通过采用负载均衡机制来让多个服务器分担负载，通过服务分级来满足不同用户的不同层次的需求。负载均衡是一种策略，它能让多台服务器或多条链路共同承担一些繁重的计算或应用服务，从而以较低成本消除网络瓶颈，提高网络的性能、灵活性和可靠性。它的优势是：提供最短的平均任务响应时间；能适于变化的负载。服务分级是指网络在传输数据流时要求满足的一

系列服务请求及实现这些请求的机制。用户可以和Web服务运行环境进行协商，需要提供什么样的服务，而Web服务运行环境通过自身的设置来满足不同的需求，从而能够为不同的用户提供不同级别的服务。

随着电子商务和电子政务的迅速崛起，基于Web的应用模式迅速发展，Web应用从局部化发展到全球化，从集中式发展到分布式，Web服务成为电子商务和电子政务的有效解决方案。从应用领域的角度来看，对Web服务性能提出了更高要求。本小节在对Web服务性能现状进行分析的基础之上，结合Web服务调用模型综合分析了影响Web服务性能的因素，并分服务传输、SOAP协议实现、Web服务运行环境三个层次分析了Web服务性能优化的技术并对这些技术的优劣进行了评价。

Web服务性能优化不是一个单一的问题。无论是在设计还是在实现的时候都需要考虑性能优化；不仅需要考虑某一种性能优化的技术，还需要从整体上去考虑如何优化性能。

4.2 基于Web服务的类型映射机制

Web服务类型映射用于解决XML表示的数据类型与目标系统的数据类型之间的转换问题。通过改善类型映射的性能，可以有效地提高报文的处理效率，从而在保证SOAP报文原有优势的同时，还能够提高Web服务性能。目前这方面的研究还比较少。

本小节在分析影响Web服务性能的因素及类型映射机制的基础之上，着眼于提高Web服务的性能，从改善Web服务的类型映射性能入手，采取WSDL文档校验机制，提出了一种可扩展的类型映射

机制，实现了其原型系统。实验结果表明：这种类型映射机制能够有效提高 Web 服务的性能，并保证了类型映射的可扩展性。

4.2.1　Web 服务与类型映射

影响 Web 服务性能的原因是多方面的，包括报文的处理以及网络传输的影响等因素。通过分析影响 Web 服务性能的主要因素，本研究认为类型映射是影响 Web 服务性能的关键因素之一。本节首先分析影响 Web 服务性能的因素，然后结合类型映射分析其对 Web 服务性能的影响。

4.2.1.1　影响 Web 服务性能的因素

基于 Web 服务的应用程序的性能通常是以它响应请求的速度来度量的。目前在基于 Web 服务的解决方案中，Web 服务性能的影响因素主要涉及以下几个方面：

①SOAP 消息的解析：SOAP 协议建立在 XML 技术的基础之上，要求编码格式是 ASCII 文本。这是使用 SOAP 的最大的优势，因为应用程序在通信之前不需要彼此协商。然而，既然编码格式是 ASCII 文本，所有的 SOAP 消息中的自描述信息都需要被转换为 ASCII 字符串的形式，传输的时候要从 ASCII 格式转换为二进制编码，会花费大量的转换代价，并且会造成网络传输的高成本，因为 ASCII 编码的消息比二进制的原始消息要大，解析它所需要花费的时间就会更长。

②Web 服务的类型映射：类型映射涉及目标系统的数据类型到 XML 类型的序列化，以及 XML 类型到目标系统的数据类型的反序列化。SOAP 消息的结构越复杂，涉及的类型越多，程序设计的对

象和 XML 元素之间的映射所需要的时间就越长。

③Web 服务安全性处理：为保证应用程序终端间的安全性，需要增加 Web 服务的一些安全处理特性，包括访问控制、Web 服务加密、Web 服务数字签名、Web 服务授权等功能，这些安全特性可能会惊人地延长处理服务请求的时间，降低 Web 服务的性能。

④网络传输以及硬件设施：网络的传输速度、路由情况、服务器的硬件配置及对请求的处理能力也会对 Web 服务性能造成重要的影响。

从以上分析可以看出，影响 Web 服务性能的因素是多方面的。类型映射是 Web 服务处理所必须涉及的一个环节，也是影响 Web 服务性能的一个关键因素。

4.2.1.2 Web 服务与类型映射

在 Web 服务请求处理过程中，无论是客户端还是服务器端，都会使用类型映射机制完成目标数据类型和 XML 类型的相互转换。以目标数据类型是 Java 类型为例，在 Web 服务调用过程中类型映射机制的工作原理如图 18 所示。

Web 服务客户端在调用一个 Web 服务时，需要使用序列化器把 Java 对象序列化成一个 XML 对象，并封装成 SOAP 报文，通过网络传输发送给 Web 服务器。

Web 服务器端在接收到 SOAP 报文后，需要寻找相应的反序列化器，把 XML 对象反序列化为 Java 对象。完成对 Web 服务的具体调用之后，服务器端使用序列化器把 Java 类型序列化成 XML 类型，并封装成 SOAP 报文，返回给客户端。客户端接收到 SOAP 响应报文，使用反序列化器把 XML 类型反序列化成 Java 类型，供应用程

序进一步处理。

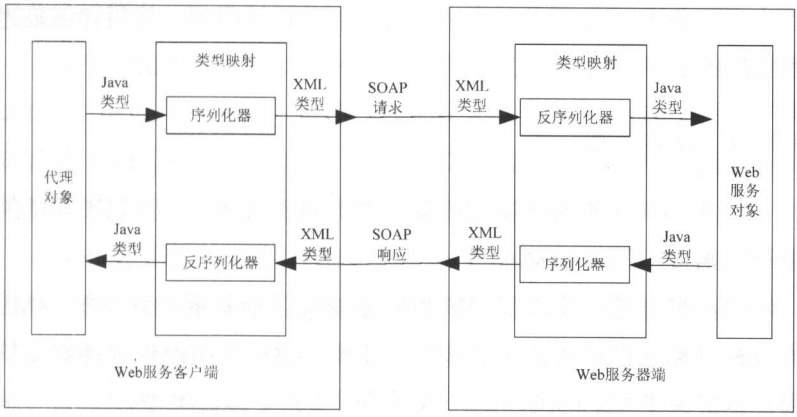

图 18　在 Web 服务调用过程中类型映射机制的工作原理

通过对类型映射机制的工作原理的分析可以看出：无论是在 Web 服务客户端还是在 Web 服务器端，类型映射在 Web 服务调用的请求和响应过程中都起着重要的作用，并且对 Web 服务性能也具有重要的影响。在现有的一些实现中，类型映射部分不仅需要完成类型映射功能，而且要进行类型验证，增加 Web 服务调用处理的时间。因此，优化类型映射处理过程，提高 Web 服务类型映射性能，是一条提高 Web 服务性能可行的途径。

4.2.2　基于 Web 服务的类型映射机制

Web 服务的类型映射机制要求保证 XML 类型和目标系统类型之间能够进行正常的互相转换。为了优化 Web 服务类型映射，基于 Web 服务应用模式，本书提出了一种 Web 服务的类型映射机制，它实现了 XML 类型和目标系统类型之间的双向转换。同时，为提高类

型映射的效率，引入 WSDL 文档校验机制来进行类型检查。下面，首先介绍基于 Web 服务的类型映射机制的设计思想，然后详细描述其实现机制。

4.2.2.1 设计思想

在基于 Web 服务的类型映射机制的设计过程中，遵循了如下的设计原则：

①可扩展性：实现类型映射的系统除了能够提供现有的 XML 类型和目标数据类型之间的转换，还要提供一种机制保证能够支持用户自定义类型之间的映射，从而能够支持更多的类型。

②性能优化：类型映射机制设计的好坏，在一定程度上影响着 Web 服务性能。在类型映射机制的设计中，分析和研究对 Web 服务性能有影响的因素，进而采取有效的优化措施。在该过程中，发现通过优化类型校验、选择适当的解析器有助于提高 Web 服务性能。此外，还需采取其他一些措施，如引入缓存机制等来优化 Web 服务的性能。

③互操作性：Web 服务能够得到广泛应用的原因之一就在于它的互操作性。类型映射作为 Web 服务的一个组成部分，同样要遵循现有的技术标准如 JAX-RPC，支持系统间的互操作性。

4.2.2.2 类型映射机制描述

为便于理解和叙述，首先给出必要的基本概念。下文中均假定目标系统为 Java 平台，目标系统类型为 Java 类型。

序列化（Serialization）：将 Java 数据类型转换为 XML 类型的过程。设 A 为任意一个 Java 语言中所定义 Java 类型，B 为任意一个

XML 类型，序列化是指具有 A 类型的 Java 对象到具有 B 类型的 XML 对象的转换η：A→B。

反序列化（Deserialization）：将 XML 类型转换为 Java 数据类型的过程，它是序列化的一个逆过程。设 A 为任意一个 Java 语言中所定义的 Java 类型，B 为任意一个 XML 类型，反序列化是指具有 B 类型的 XML 对象到具有 A 类型的 Java 对象的转换η：B→A。

Web 服务类型映射（Web Services Type Mapping）：Web 服务中的类型映射是用于实现在 Java 数据类型和 XML 类型之间的转换，从而能够将 Java 对象转换为 XML 对象，并在 SOAP 消息中进行传输。它包括两个过程：序列化和反序列化。设 A 为任意一个 Java 语言中所定义的 Java 类型，B 为任意一个 XML 类型，Web 服务类型映射是指具有 A 类型的 Java 对象到具有 B 类型的 XML 对象的互相转换η：$A \rightleftharpoons B$。

类型映射项（Type Mapping Item）：用于描述类型映射中 XML 类型、Java 类型、序列化器和反序列化器的数据结构，可以定义为一个四元组：< XMLType，JavaType，Serializer，Deserializer>。其中，XMLType 是 QName 形式表示的 XML 数据类型名称，如 xsd：string、xsd：int 等；JavaType 是相对应的由 Java 语言所规定的 Java 数据类型，如 java.lang.String、java.lang.int 等；Serializer 是序列化过程中用于把 Java 对象序列化成 XML 实例的序列化器；Deserializer 是反序列化过程中用于从 XML 实例重新构造 Java 对象的反序列化器。

根据 Web 服务应用模式和 JAX-RPC 规范，本书提出一种 Web 服务的类型映射机制（见图 19），保证 XML 类型和 Java 类型之间的相互转换，并通过引入 WSDL 类型校验机制来优化类型映射的效率。

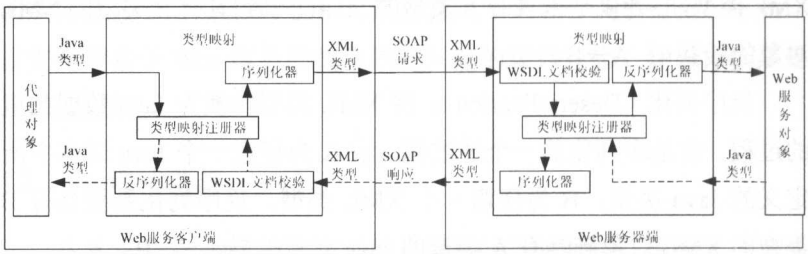

图 19　基于 Web 服务的类型映射机制

类型映射注册器提供了对类型映射项的管理功能，既支持已有的类型，也支持用户自定义类型，保证了类型映射的可扩展性。

在 SOAP 报文反序化的过程中，引入 WSDL 文档校验机制，用于进行类型校验以及确定 XML 类型的反序列化器。通过使用 WSDL 文档进行类型校验，可以解决以下几个方面的问题：

①及早进行类型检查：通过 WSDL 文档进行 XML 类型验证，检查 SOAP 报文中的 XML 类型是否合法，以及 WSDL 文档中是否有 Web 服务所进行操作的方法。如果没有通过类型检查，可以直接返回含有错误信息的报文，而不必在反序列化的过程中才发现问题。

②优化反序列化器的查找过程：仅根据 XML 类型在类型映射注册器中查找对应的反序列化器，需要先查找相对应的 Java 类型，然后才能根据 XML 类型和 Java 类型确定类型映射项，进而确定相应的反序列化器。引入 WSDL 文档校验之后，通过 WSDL 文档可以直接确定 XML 类型所对应的 Java 类型，能够很快查找到对应的反序列化器，从而优化了反序列化器的查找过程。

③支持 Java 操作重载：通过获取报文中的 Web 服务操作名称以

及参数类型，与 WSDL 中所描述的操作进行匹配，确定当前所进行的操作，支持 Java 类中同名但参数不一致的多个方法，从而解决了 Java 操作重载的问题。

4.2.3 WSTM 系统的设计与实现

根据上述对 Web 服务与类型映射的分析，结合第 3 节提出的 Web 服务类型映射机制，设计并实现了 Web 服务类型映射子系统 WSTM。下面将从系统的总体结构设计、系统主要组成部分的工作原理和实现机制等方面来进行介绍。

4.2.3.1 系统的总体结构设计

考虑到类型映射的可扩展性、互操作性和性能优化，本书设计并实现了 WSTM 系统。WSTM 系统主要由类型映射管理器、类型映射注册器等部分组成（见图 20）：类型映射管理器负责处理请求报文及响应报文，使用 WSDL 文档校验器完成对 SOAP 报文中 XML 类型的校验，调度类型映射注册器，为 XML 类型确定反序列化器以及为 Java 类型确定序列化器；此外，类型映射管理器还提供了 WSDL 文档解析功能。类型映射注册器负责管理已注册的类型映射项，提供了注册/反注册类型映射项、查找序列化器/反序列化器的功能。

4.2.3.2 类型映射管理器

类型映射管理器是 WSTM 的主要组成部分，它负责调度类型映射注册器以及 WSDL 文档校验器，确定合适的序列化器/反序列化器，完成序列化/反序列化功能。它的工作过程如下：

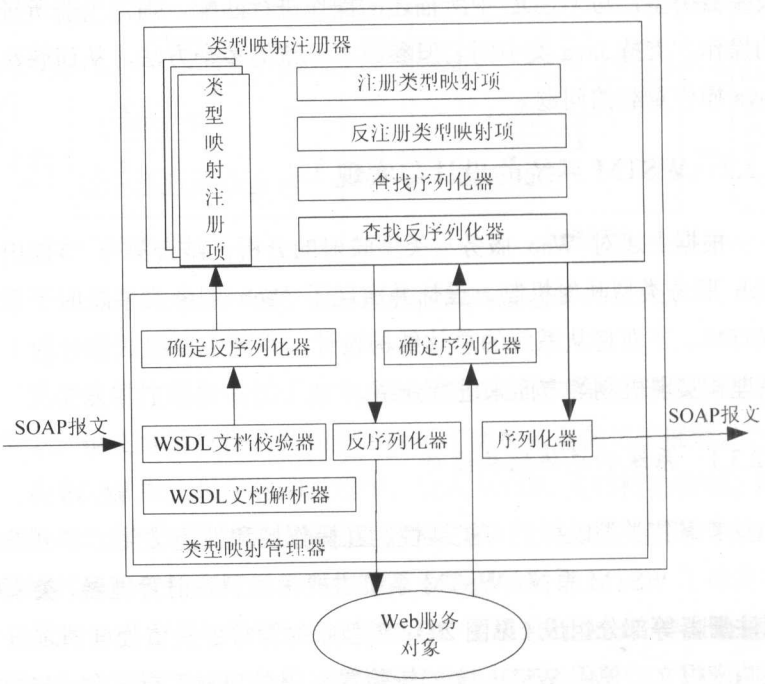

图 20　WSTM 的系统体系结构

（1）收到 SOAP 报文

- 根据所调用服务的名称，在缓存中查找该服务所对应的 WSDL 对象。类型映射管理器采取了缓存 WSDL 文档对象的机制，减少因读取 WSDL 文档而带来的新的开支。如果缓存中无法找到该 WSDL 对象，则通过 WSDL 文档解析器读取解析该服务的部署描述文档，生成该服务的 WSDL 对象。
- 利用 WSDL 对象，校验 SOAP 报文是否是合法的报文，如果是非法的报文，则直接返回包含错误信息的报文；此外根

据 SOAP 报文中的 XML 类型，获取其相对应的 Java 类型。

- 利用所获取的 XML 类型和 Java 类型，在类型映射注册器中查找反序列化器。
- 反序列化器反序列化 SOAP 报文，生成 Java 对象，并调用后端所实现的服务。

（2）返回 SOAP 报文

服务调用完毕，服务器端要把调用结果即 Java 对象序列化成 SOAP 报文，并返回给客户端。

- 类型映射管理器根据 Java 对象的 Java 类型，在类型映射注册器中查找序列化器。
- 利用找到的序列化器把 Java 对象序列化成 XML 实例，封装成 SOAP 报文，返回给客户端。

（3）客户端的处理

客户端需要按照与服务器端一致的方式来处理接收到的 SOAP 报文。

4.2.3.3　类型映射注册器

类型映射注册器负责管理在系统中注册的类型映射项，提供了注册、反注册、查询类型映射项的功能，并且提供了类型映射项动态注册功能，以便对自定义类型进行支持。为了保证系统的互操作性，系统映射注册器遵循 JAX-RPC 规范。下面从类型映射项动态注册、序列化对象缓存机制、对象序列化算法等方面进行阐述。

4.2.3.4　类型映射项动态注册

类型映射注册器内建支持一些类型映射项。为了对自定义类型

进行支持，类型映射注册器提供了类型映射项动态注册功能。

如果Web服务的实现涉及自定义的Java类型或者XML类型，那么用户需要在该服务的部署描述文档中描述类型映射项的信息，如XML类型、对应的Java类型、进行转换所需的序列化器、反序列化器等。Web服务部署完成之后，系统会通过读取该服务的部署描述符，加载自定义的类型映射项，从而实现对自定义类型的支持。如果自定义的类型映射项与已有的类型映射项中的Java类型和XML类型一致，但是它们的序列化器和/或反序列化器不一致，那么会用自定义的类型映射项覆盖原有的类型映射项。

4.2.3.5 序列化对象缓存机制

Java类型可以分为简单类型和复杂类型，其中复杂类型包括集合类型如HashMap、JavaBean、自定义类型等。复杂类型中的子元素可能会包含多个简单类型或复杂类型，从而形成一个嵌套的结构。复杂类型是使用频率很高的一种类型，这样在序列化一个复杂类型对象的时候，就可能对同一种类型的对象进行多次序列化，从而造成了较大的时间开支。

因此针对这种情况，在序列化器的实现过程中，为了能够提高序列化的效率，采取了序列化对象缓存机制，具体描述如下：

①在序列化器的消息上下文中注册已经序列化的对象以及该对象对应的XML实例。

②在序列化一个对象的时候，首先根据该对象的ID在消息上下文中进行搜索，判断该对象是否已经注册。如果已经注册，说明前面对该对象已经进行了序列化，无须执行序列化的过程，直接生成该对象的XML实例，否则序列化该对象，并且把该对象和生成的

XML 实例在消息上下文中进行注册。

通过采用序列化对象缓存机制，避免了对同一对象进行多次序列化，有效地减少对象序列化的开支，提高对象序列化的效率，尤其在序列化对象是复杂类型的情况下。

4.2.3.6　对象序列化算法

序列化器的功能是序列化 Java 对象，生成 XML 实例。对象序列化算法将消息上下文中的 Java 对象返回值列表序列化，将序列化结果存入消息上下文的返回消息中。下面简要介绍该算法：

①获取消息上下文中的 Java 对象返回值列表，这个列表的类型为 Object[]。

②根据消息上下文中的 WSDL 定义确定返回值的 XML 类型。

③根据①②中结果以及类型映射注册器中的信息，确定所要使用的序列化器。

④调用上述序列化器的序列化方法，生成 SOAPElement 元素，保存到消息上下文的响应消息中。

⑤循环，直到处理完所有参数。

其间如果发生以下情况，则抛出类型映射异常：

①期望的序列化器不存在。

②序列化过程失败。

4.2.4　实验结果及分析

本小节进行两组实验分析，分别进行简单类型和复杂类型测试，比较类型映射机制引入前后对系统性能的影响，以及在请求并发数不同的情况下对 Web 服务性能的影响情况。

实验基于局域网环境，由一台机器作为服务器，一台机器作为客户端模拟请求。配置如下：

服务器：Intel Pentium 4 2.0GBytes，512MBytes 内存，10/ 100MBytes 自适应网卡。

客户端：10MBytes 网卡，请求产生工具采用 MS.NET 平台提供的 ACT。模拟多个并发线程发送请求，每个请求接收到响应后发送下一次请求。

两组实验的用例分别为简单类型服务用例和复杂类型服务用例。

简单类型服务用例：采用的 Web 服务用例是执行 Echo 操作，功能是服务器端直接返回客户端发送的请求内容。

复杂类型服务用例：采用的 Web 服务用例是获取订单操作，功能是服务器端根据客户端发送的客户信息，返回相应的订单信息。

两个服务用例均部署在服务器上，供客户端调用。

图 21 是请求并发数与每秒平均请求数之间的关系，图 22 是请求并发数与类型映射所占百分比之间的关系。其中 S1、S2 分别代表类型映射机制引入前后的简单类型；C1、C2 分别代表类型映射机制引入前后的复杂类型。

从而可以看出，引入新的类型映射机制之后：

①每秒平均请求数比之前有较大幅度的提高；而且复杂类型比简单类型提高的幅度要大，说明对复杂类型的影响较大。

②类型映射在 Web 服务调用过程中所占时间百分比有所下降，说明类型映射部分占用的时间较少；同样对复杂类型的影响大于对简单类型的影响。

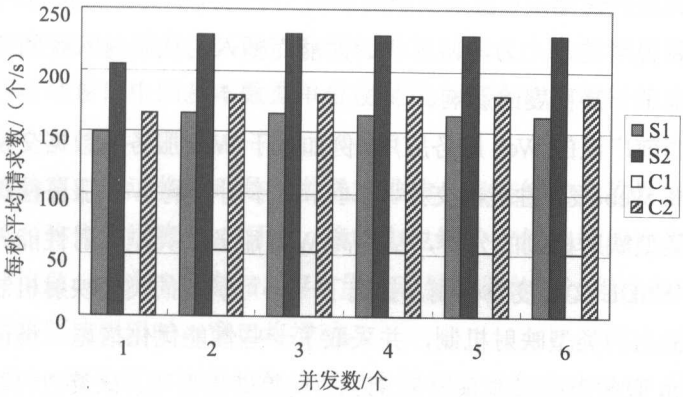

图 21　请求并发数与每秒平均请求数之间的关系

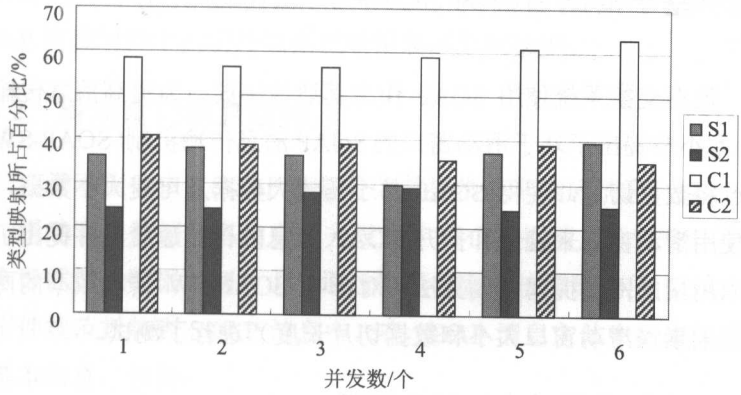

图 22　请求并发数与类型映射所占百分比之间的关系

③请求并发数对性能影响较小。

从上面的分析可以看出，引入新的类型映射机制，不仅降低了类型映射子系统在 Web 服务调用中所占的百分比，而且提高了 Web

服务调用的性能；对复杂类型的影响大于对简单类型的影响。此外，必须要说明的是，为测试模块的性能而插入的代码对系统的整体性能造成了一定程度的影响。

当前广泛的 Web 服务应用。例如基于 Web 服务的数据交换迫切需要高 Web 服务性能的支持。本小节基于对影响 Web 服务性能的因素及类型映射机制的分析，从提高 Web 服务的类型映射性能入手，建立 WSDL 文档校验机制，提出了一个可扩展的类型映射机制。基于所提出的类型映射机制，并采取了一些性能优化措施，设计实现了 Web 服务类型映射系统 WSTM。实验结果表明，该类型映射机制在实现类型映射的功能的同时，还能够有效提高 Web 服务性能。

4.3 基于滑动窗口的 SOAP 消息传输机制

数据交换系统使用 SOAP 作为其通信协议。为提高消息传输效率，本小节提出了基于滑动窗口的 SOAP 消息传输机制（SOAP-SW），通过将数据切片，使得 SOAP 易于传输大数据及可变大小数据，通过使用滑动窗口来控制和提升 SOAP 消息的传输速度，并提出了控制应用层的网络拥塞控制算法，对影响 SOAP-SW 传输效率的两个主要因素（滑动窗口大小和数据切片长度）进行了测试。

4.3.1 SOAP 滑动窗口工作原理

针对 SOAP 消息传输效率低的问题，目前已经有不少改进方案，如：通过改进 SOAP 绑定来提高传输效率；根据不同的应用场合，采用适合的 SOAP 消息编码规则；减少 SOAP 消息构造和解析；采用压缩技术，对 SOAP 消息体积进行压缩等。

本小节提出的 SOAP-SW，是在 SOAP 消息传输中引入滑动窗口机制，将 SOAP 消息待传输的数据切片，以切片作为 SOAP 消息的附件，滑动窗口中的基本数据单位便是这种带有切片附件的 SOAP 消息。

滑动窗口的工作原理是：发送方维护一组连续的允许发送的数据信息，称为发送窗口；同时，接收方也维护一组连续的允许接收的数据信息，称为接收窗口。窗口用来存储数据传输过程中的数据，当数据需要被更新时，窗口将滑动。

SOAP 滑动窗口与其他滑动窗口不同的地方在于：SOAP 滑动窗口中存储的是 SOAP 消息，滑动窗口用来控制 SOAP 消息发送和接收的速度，进行网络拥塞控制，并保证 SOAP 消息中附件数据的可靠有序的传输，当传输结束之后，接收方滑动窗口将接收到的所有 SOAP 消息附件中的切片数据重新组装成原始数据。

4.3.2　滑动窗口消息结构

在滑动窗口中，SOAP 消息分为数据消息和响应消息。

滑动窗口数据消息由发送方发出，每个数据消息中包含一个或多个数据切片数据，切片大小通过协商确定，按照约定的 SOAP 消息附件规范封装到 SOAP 消息中。在 SOAP 消息主体中指出该数据的基本信息，包括：

①SOAP 消息编号。

②待发送的数据编号。

③切片编号。

④数据切片个数。

⑤每个数据切片的大小。

⑥所采用的 SOAP 附件封装规范。

⑦是否是当前正在传输的数据的最后一个数据切片。

⑧是否是最后一个数据。

SOAP 滑动窗口响应消息由接收方发出，是对发送方所发出的数据消息的确认。响应消息的 SOAP 消息主体中包含如下信息：

①SOAP 消息编号。

②已确认的数据编号。

③已确认的数据切片编号。

④是否是当前正在传输的数据的最后一个数据切片。

⑤是否是最后一个数据。

⑥响应消息类型。

⑦具体响应信息。

4.3.3 滑动窗口工作流程

基于滑动窗口的 SOAP 消息传输方法其整体过程主要如下：

①发送方和接收方通过三次握手建立连接，该方式可以采用事先协商的传输协议和端口来实现。在三次握手中，需要确定如下内容：SOAP 附件封装规范、传输协议、滑动窗口大小、数据切片大小、超时时间间隔、超时重传次数上限、单个 SOAP 消息中数据切片个数等信息。

②根据三次握手建立的约定，分别对发送方和接收方进行初始化，包括初始化滑动窗口、启动发送方和接收方的定时器等。

③发送方按照约定的数据切片大小读取数据，将数据切片封装成 SOAP 滑动窗口数据消息，并放入滑动窗口中进行发送，在发送结束之后等待接收方的响应消息。

④接收方接收 SOAP 滑动窗口数据消息，解析该 SOAP 消息，并根据解析结果，选择将数据放入滑动窗口中还是丢弃；对于放入滑动窗口中的数据，需要再判断数据是否接收完毕，滑动窗口是否滑动，并向发送方发送 SOAP 滑动窗口响应消息。

⑤发送方接收到 SOAP 滑动窗口响应消息后，解析 SOAP 消息，并根据解析结果判断数据是否发送完毕，如果没有发送完毕，判断是重发数据消息还是将滑动窗口进行滑动以读入后续数据并发送。

⑥发送方、接收方不断重复上述第④～⑤步直到数据发送完毕。

4.3.4　应用层拥塞控制算法

基于滑动窗口的 SOAP 消息传输方法能够有效提高传输的速度，然而，它也存在一定的弊端，例如当向网络中注入数据的速度过快时，容易造成网络拥塞。TCP 协议本身具有拥塞控制机制。在 TCP 的拥塞控制中，一般采用：慢启动、加速递减以及拥塞避免三种技术。但是由于 TCP 属于传输层协议，在应用层，其所处协议栈层次较高，如果仍然采用 TCP 中使用的三种技术，频繁改变窗口大小，反而会导致效率低下。因此，本小节在基于滑动窗口的 SOAP 消息传输机制中，提出了一种应用层拥塞控制机制，从而保证 SOAP 消息传输的畅通。具体方法如图 23 所示。

①在发送方和接收方通过三次握手建立连接时，确定双方可以接受的发送窗口最大值（MAX），并且设置发送窗口的大小（S）= MAX，接收窗口大小（R）= MAX。

②按照设置的发送窗口大小，使用滑动窗口进行 SOAP 消息传输。

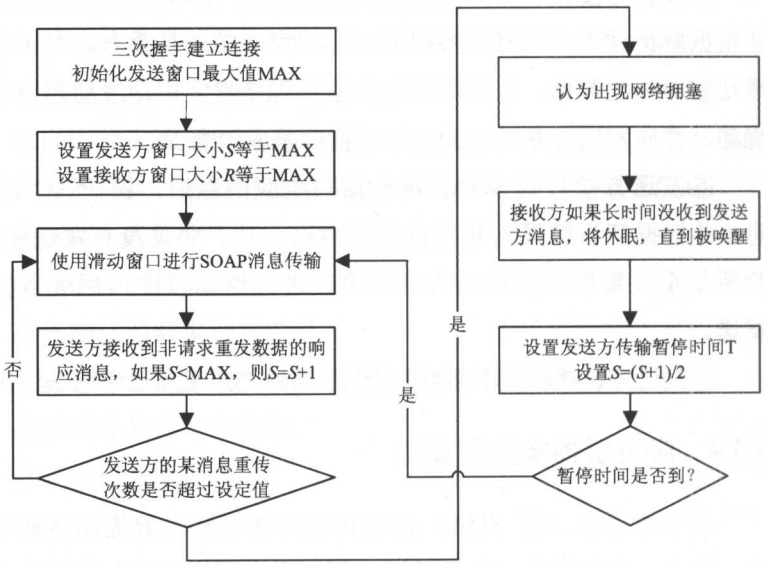

图 23　拥塞控制流程

③发送方每接收到一个非请求重发数据的响应消息的同时，判断此时发送窗口大小（S）是否小于发送窗口最大值（MAX），如果小于发送窗口最大值，则 $S=S+1$；否则，保持发送窗口大小不变。

④在传输过程中，如果发送方发出的 SOAP 消息连续重传次数小于等于某个经验值（例如 3 次），则认为网络传输通畅，重复执行第②～④步。

如果发送方发出的某 SOAP 消息连续重传次数大于经验值（例如 3 次），则认为出现网络拥塞，数据传输暂停 DELAY_T 时间，设置 $S=(S+1)/2$；这时接收方如果长时间没有收到消息，将休眠，直到被发送方唤醒。

⑤判断暂停时间是否到，如果没有到，继续暂停；如果时间到，则继续执行第②～⑤步直到数据传输完毕，传输结束。

在该拥塞控制方法中，DELAY_T 是数据传输暂停时间，取决于当前网络的拥塞时间。如果接收方长时间没有接收到发送方的数据，将进入睡眠状态或者终止状态，然后由接收方唤醒。

4.3.5　影响 SOAP-SW 传输效率的主要因素

在 SOAP-SW 中，影响传输效率的主要因素如下：滑动窗口大小 S（发送方窗口大小 Ssender，接收方窗口大小 Sreceiver）、数据切片长度 L 及网络传输协议。由于网络传输协议与其自身及其实现相关，这里不予讨论。

Ssender 由数据发送和接收双方在建立连接时商定，通信过程中，发送方和接收方也可以根据网络带宽、报文往返时延或者自身软硬件环境，协商调整窗口大小以及数据切片长度。Sreceiver 由接收方自己决定。当 Sreceiver＜Ssender 时，容易使接收到的数据经常超出接收窗口范围，从而造成数据频繁丢弃；相反，如果 Sreceiver＞Ssender 时，虽然可以缓存更多的数据消息，但会造成过多内存开销。因而 Ssender 和 Sreceiver 的值最好相等或者相近。

设计了多个测试用例，测试 S 和 L 对传输效率的影响。测试采用滑动窗口大小从 1 到 50，滑动窗口和接收窗口大小相等。切片大小由于受到内存大小的制约，切片长度实际测试范围从 20kBytes 逐渐增加到 10MBytes。

图 24 为测试得到的传输时间与滑动窗口大小关系图，测试使用 HTTP 作为传输协议，分片长度为 800 kBytes，图中三条曲线分别是 RTT＜1 ms、RTT=200 ms 以及 RTT=400 ms 时，滑动窗口由 1 增加

到50的测试结果。

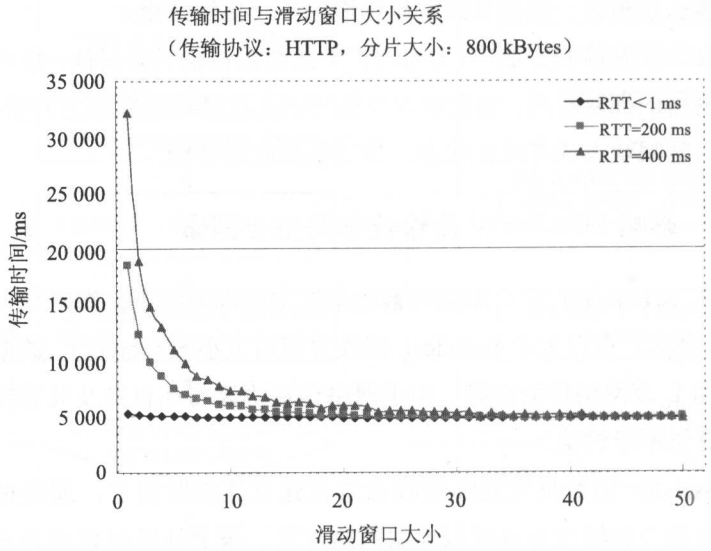

图24 传输时间与滑动窗口大小的关系

从图24可以看出，滑动窗口越大，传输时间越短，滑动窗口增大到一定程度时，传输所用时间达到一个极值，几乎不再减少，这是由于窗口越大，消耗的资源也越多的原因造成的，窗口大小增大到一定程度时，内存和CPU处理速度成为性能瓶颈。当前测试环境下，窗口大小为5～10时，传输效率已经提高到一个比较佳的状态。此外，滑动窗口能够消除由于RTT造成的传输时间的差异，当窗口大小增加到一定程度时，不同RTT条件下的传输时间几乎相同。

图25为测试得到的传输时间与分片长度的关系图，测试采用的RTT=100ms，窗口大小为6，图中两条曲线分别是HTTP和TCP两种传输协议情况下，数据切片长度由20 kBytes逐渐增加到

10 MBytes 绘制的测试结果。

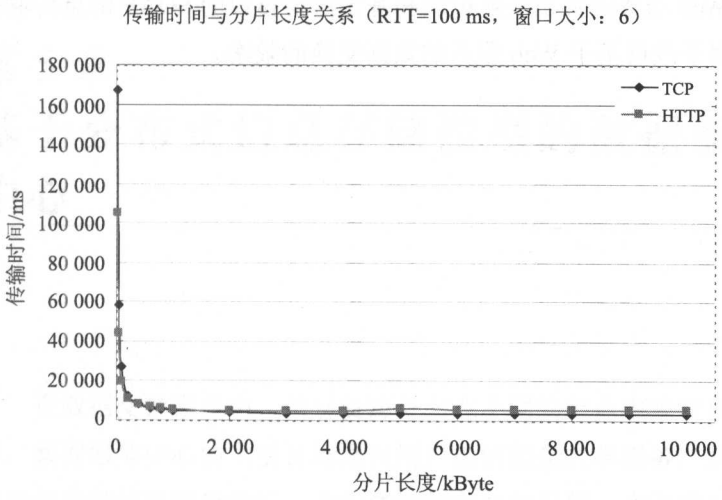

图 25　传输时间与分片长度的关系

从图 25 可以看出，数据切片长度越大，传输时间越短，数据切片长度增到一定程度时，传输所用时间达到一个极值，几乎不再减少。显然，切片长度受到内存、传输并发数、CPU 处理能力等的限制，因此必须根据实际需要选择合适的值。当前测试环境下，数据切片长度增加到 400 kBytes 时，传输效率就能提高到一个较好的状态。

4.4　本章小结

本章首先分析了基于 Web 服务的数据交换中影响 Web 服务性能的因素，从服务传输层、SOAP 协议实现、Web 服务运行环境三个

层次进行了分析并提出了相应的优化方案；在此基础之上，设计了基于Web服务的类型映射机制和基于滑动窗口的SOAP消息传输机制来用于改进基于Web服务的数据交换的效率。

第 5 章
基于分布式信息存储模型的数据服务中心

在数据交换系统中，信息资源分布在不同的节点中来管理和维护，数据服务中心的主要目标是向用户提供信息共享服务，从而更好地对资源信息进行描述、方便资源的发布与发现。本章在对数据交换中数据服务中的可扩展性、发现效率和有效性进行分析的基础上，设计了一个分布式信息资源存储模型，讨论了注册中心的组织，提出了信息动态注册与更新机制，设计了一个分布式信息搜索算法，并设计实现了基于 UDDI 的数据服务注册中心。

5.1 数据服务分析

在广域网中，信息资源分布在不同的节点中来管理和维护，数据服务中心的主要目标是向用户提供信息共享服务，从而能够更好地对资源信息进行描述、从而有利于资源的发布与发现、提供对资源快速下载或显示的支持等。为了满足这些目标，本节对在数据交换中数据服务的关键问题进行了分析，并提出了可能的解决策略。

①可扩展性：在广域网中，无论面向电子政务或者是面向电子商务领域，信息量巨大，信息种类繁多，采用单一的目录服务器来存放所有共享数据的相关信息会存在单点失效和服务器负载过大的问题，而如果进行统一管理则使得成本增加、灵活性变差，因此本研究考虑设计一种混合式结构模型来提高可扩展性。

②效率及准确度：数据服务中心的主要目的便是优化数据存储过程，提高存储效率，因此，考虑设计一种高效的信息资源组织方式，并设计相应的资源搜索算法来提高搜索效率及准确性。

③有效性：信息资源是动态变化的，需要设计一个信息资源动态注册与更新机制来保证资源的实时有效性，同时减少由于信息更新而造成的消息负载。

通过对上述关键问题及可能的解决策略的分析，结合广域网下数据交换的特殊性，在 Web 服务、P2P（Peer to Peer computing，对等计算）等技术的基础上提出了一种基于混合式结构的分布式信息存储模型，并且在此基础上采用树作为存储模型来组织超级节点层的注册中心，从而有利于资源的发布和发现，提高可扩展性及灵活性，并基于 UDDI 技术设计实现了数据服务注册中心。根据上述设计思想，本书设计了几个关键技术来作为数据服务中心效率、准确度、有效性等问题的解决方案：第一，设计了分布式信息存储模型，解决了可扩展性问题；第二，设计了一种信息资源动态注册与更新机制为信息资源提供动态注册与更新，保证了信息资源的有效性；第三，设计了适合于该模型的分布式信息资源搜索算法，提高了系统搜索效率和准确性；第四，设计实现了一个基于 UDDI 标准的数据服务注册中心。下面将讨论数据服务中心中相关的关键技术。

5.2　分布式信息资源存储模型

集中式存储模型是一种常用的信息资源发现与共享资源的方式，如 Web 搜索引擎，在这种存储模型中，所有资源的信息都存放到单一的服务器中。集中式存储的优势在于：实现相对简单，易于统一管理。但是在广域网电子商务及电子政务应用中，把资源信息集中存放在一个服务器中会有以下缺点：①在服务器存放的信息量过载，会影响访问效率，并易导致单点故障，从而难以适应大规模的数据访问；②由于信息管理和维护涉及多个不同的部门，为自治管理带来很多不便。

分布式的信息资源存储是信息共享领域发展的重要趋势。目前，对等计算系统在文件共享方面得到了广泛应用，证实了分布式存储结构的扩展性，但同时也产生了一些新的问题：基于泛洪的非结构化对等计算系统，信息资源分散存放在各对等节点上，易于维护，但可扩展性差，因为资源的发现只能依靠泛洪来处理，搜索效率较低，并且大量的消息负载会导致严重的网络堵塞，它仅适合于小范围的信息共享；采用基于 DHT 的结构化对等计算系统，在资源定位时不需要信息泛洪，而是通过利用较少的路由信息便可直接定位到目标节点，它的可扩展性好，信息资源的搜索效率高。但由于实现算法复杂且不支持复杂条件查询，不适合于信息资源动态变化的场景，且不利于对资源的自治管理。

本小节结合集中式存储在小范围条件下的高效性和易管理性以及分布式存储的可扩展性等优势，设计出一种适合于广域网下数据交换应用的具有可扩展性的分布式信息资源存储模型，用来组织网

络中的各个节点，以保证信息资源组织的高效性。如图26所示，网络中的节点被赋予信息提供者、信息使用者和注册中心三种角色。从网络逻辑层次上，该模型可以划分为资源层、普通节点层和超级节点层。

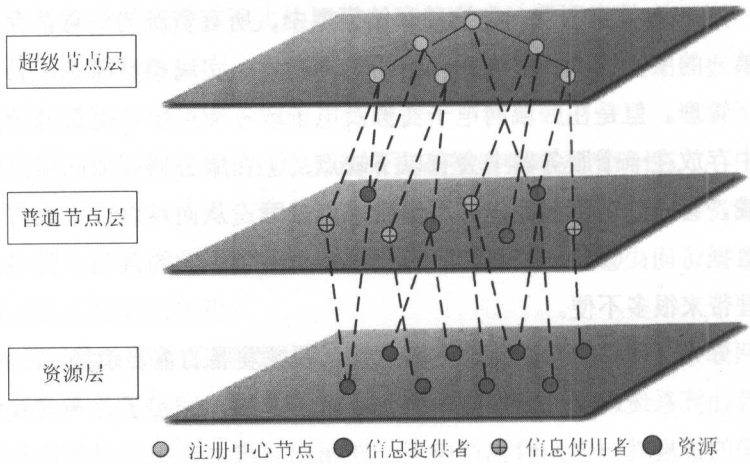

图26 分布式信息资源存储模型

资源层涵盖了数据交换中以共享、传递、交换为目的的信息资源。为了能够有效地共享这些资源，资源的使用者必须了解资源的相关信息，即资源的元数据信息，上层的信息提供者负责对元数据信息的收集。

普通节点层包括各种信息提供者和信息使用者，统称为客户端节点。信息提供者负责从资源层中收集资源的相关信息，按照统一的信息模型提供给注册中心。信息使用者需要查询某项资源时，直接向注册中心提出查询请求，在获得查询结果后和信息提供者建立连接，从下层的资源层中取得所需资源。

超级节点层汇集了网络计算能力比较强的节点，统称为注册中心。在该模型中，将这些分布在不同级别的注册中心组织成树形结构，每个注册中心负责管理某个区域的节点，维护本区域的信息资源，并与其邻近的部分客户端节点一起构成一个自治的簇。这种组织方式既充分考虑了现实物理组织结构的构成，又能够保证各个区域的自治性。

本系统设计的分布式结构与 C/S 结构的区别在于：C/S 结构中所有共享的信息资源都必须存放于服务器端，由服务器端提供资源的下载；在该模型中，具体的资源内容始终存放于本地机器上，注册中心只是维护与资源相关的元数据信息。由于该模型具备了混合式对等计算结构的优点，从而可以有效地避免单点失效问题，具有较高的可扩展性和灵活性；同时，设计了超级节点，在一定程度上能够提高网络负载平衡，获得较高的稳定性和效率。

5.3　注册中心的组织

在分布式信息资源存储模型中，将数据交换网络中的节点划分为多个区域，每个区域由计算能力较强的超级节点管理和维护，这些超级节点通称为注册中心。在超级节点层中，注册中心的组织和管理会成为影响搜索效率的关键因素。本书中，考虑到树形结构的高效遍历算法以及与电子政务应用中组织结构的相似性，本书采用树形存储模型来组织注册中心。下面将主要讨论注册中心的加入与退出机制。

系统管理员负责配置节点的父节点信息，在父节点确定之后，即可以确定一棵目录树的结构。

5.3.1 注册中心的加入

一个注册中心加入目录树的过程是：

①读取本地配置服务的父节点信息，通过与父节点的交互，完成新节点的注册。

②假如父节点不在逻辑网络中，该节点将按照一定的时间间隔不断尝试与父节点建立连接，一旦父节点加入到网络中，子节点便能够与父节点建立通信。

③当父节点接收到子节点的加入信息后，便会将子节点的信息加入到相应的数据表中。

随着时间增加，各注册中心能够动态获取和更新其子节点的活动列表，从而帮助注册中心在资源搜索时决定向哪些子节点转发查询请求。

5.3.2 注册中心的退出

在数据交换中，一般都把计算处理能力较强的节点作为服务器，由这些节点承担路由转发和区域管理等责任。这些节点被选择作为注册中心，假定这些节点不会轻易退出 Web 应用系统平台，因此本书暂不考虑注册中心被迫退出的情况。如果出现了某个注册中心退出而其他注册中心并不知情的情况，可以通过发送试探性消息来判断其状态，如果发送不成功则默认其已经退出，在其区域内的客户端节点将不再参与资源共享。

5.4　信息资源动态注册与更新

信息的注册只是信息使用的第一步，只有经过注册的信息资源才能够被请求者使用。信息提供者负责在指定的注册中心注册共享的资源信息。在资源注册之后，由于情况发生变动，也可能需要对资源信息进行删除或修改，即需要信息更新。在这里需要考虑的是，如何动态获取资源的更新信息，从而保证信息的有效性，避免资源的变化情况不能及时反映到注册中心。

从以上分析可以看出，由于资源的不断变化可能会导致信息的动态注册、更新和删除等情况。为保证资源信息的有效性，减少更新消息造成的负载，设计了信息资源动态注册与更新机制：

①客户端按照一定的时间间隔，周期性地检查资源信息是否改变。

②如果资源加入、更新或退出，客户端自动获取资源的更新信息，将它们发布到本区域的注册中心节点，从而保证该区域的注册中心维护的是最新的信息。

③注册中心一旦接收到来自客户端的更新消息，它会根据消息类型进行相应的处理，并将注册或更新的元数据信息缓存到数据库中。

由于注册中心维护了全局的索引信息，因此资源信息的变化会引起索引信息在目录树中沿叶子节点到根节点路径的更新消息的发布，虽然这与完全复制策略中需要在所有的注册中心之间保持同步相比，已经很大程度上减少了更新消息负载，但是如何进一步减少冗余信息而同时保证信息的一致性仍然是需要考虑的问题。注册中

心层级越高，其连接的子节点数目就会越多，从而会导致接收到索引更新消息的次数越频繁，从而会导致性能降低。为了改善系统性能，采用了一种非统一信息分发机制来动态更新索引信息：当注册中心节点接收到来自子节点的索引更新消息时，它不会立即将这些更新消息转发到其父节点，而是根据缓存的信息对这些更新消息进行过滤，以更低的更新频率将筛选后的部分更新消息转发给其父节点。

5.5 分布式信息资源搜索算法

资源发现与定位是数据服务中心的基本功能。对于分布式信息存储模型而言，资源信息可能存储在多个不同的注册中心上，用户无法知道满足要求的信息存储在哪个节点上。因此资源发现的过程便是将用户的查询请求发送到合适的注册中心节点并找到合适的资源的过程。资源发现与定位的效率会严重影响数据服务中心的性能，因此采用什么样的查找策略以便能够更快地返回最合适的资源是必须解决的关键问题之一。

用户首先将查询请求提交到本区域的注册中心节点，由该注册中心节点负责超级节点层的资源发现，并最终返回合适的资源。对于该注册中心节点而言，资源发现的过程可以分为两步：

①在本地目录服务器进行信息查询。

②在目录树的其他注册中心节点之间进行搜索，即向其他注册中心节点转发查询请求并收集查询结果。

本地信息查询主要是根据用户提交的查询请求中的限定条件，在本地目录服务器上进行查询操作，通过资源的选择和匹配来获取

满足用户需求的合适的资源信息。

当注册中心完成本地信息查询之后，还需要进一步向其他区域的注册中心节点转发查询请求并收集查询结果。利用树形存储结构的特点，设计了基于索引信息的定向泛洪算法来实现分布式信息资源搜索过程。该算法的设计思想在于：向子节点转发查询请求的时候并不是随机泛洪，而是根据本地的缓存信息和查询范围来转发查询请求，从而保证所有被转发的节点都能够根据有用的索引信息用于指导定位过程。因此，与其他随机泛洪算法相比，该算法更加有效。该算法的算法流程如图 27 所示。根据需求的不同，注册中心节点可以灵活地将查询的范围控制在本子树、其他子树或全树。发起查询请求的注册中心节点在转发查询请求之后，将会在限定的时间内从其他区域的注册中心处收集响应信息，并将查询结果转换成统一格式，最后返回给客户端节点。同时，通过增加缓存管理使得在查询时首先查询本地缓存信息以提高搜索效率。

为了验证基于索引信息的定向泛洪算法的有效性，设计了仿真实验模拟注册中心节点的树形层次结构以及资源的发布和查询过程，并与树形结构下的其他随机泛洪算法进行比较和分析。在仿真试验中，设计了以下两个度量参数：

①平均查询转发次数：在一次查询中，查询请求被转发的注册中心节点数，即消息转发的跳数，这是树形结构下影响负载的主要因素。

②查询命中率：找到有效而且符合条件的资源的请求个数占总发起请求个数的比例。

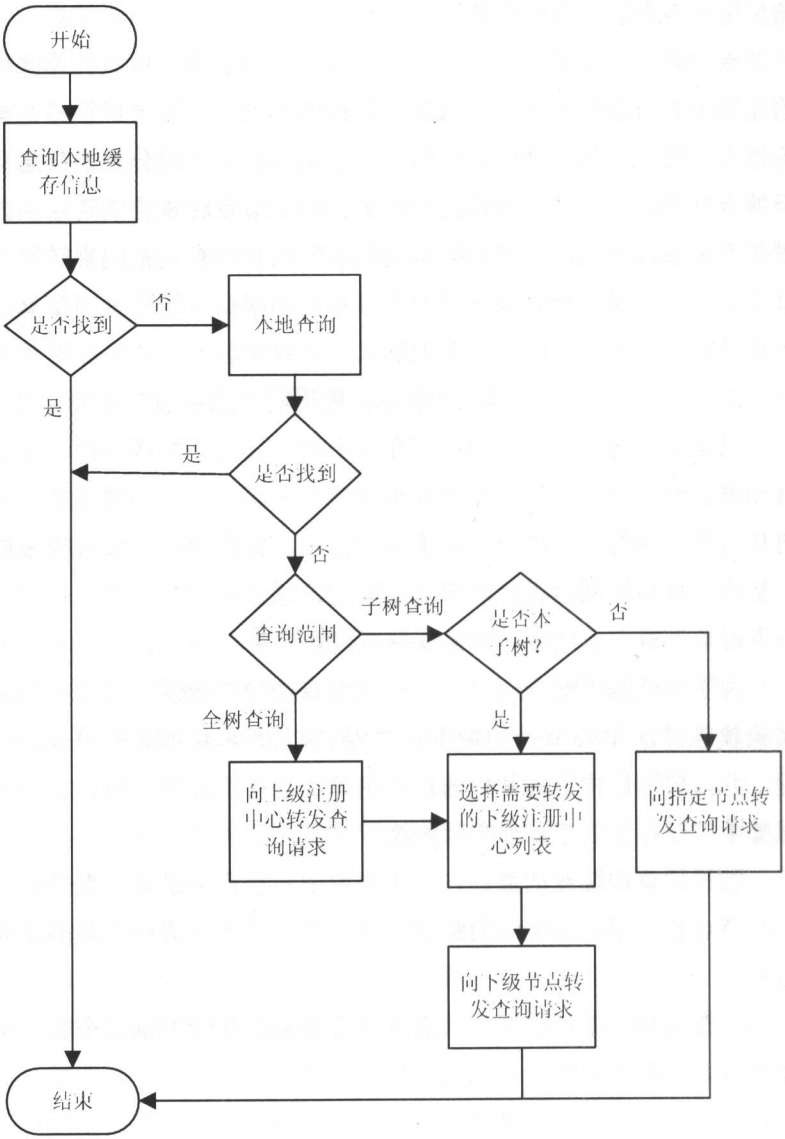

图 27　分布式信息资源搜索算法

图 28 描述了随着注册中心节点数的增加，三种算法对平均查询转发次数的影响，它反映了查询消息的负载。实验结果表明：随着 p 值的增大，树形结构下的随机泛洪算法被转发的节点个数越多，而基于索引信息的定向泛洪算法的平均查询转发次数最少，不会随注册中心节点数的增加而增加，因此造成的查询消息负载较少。

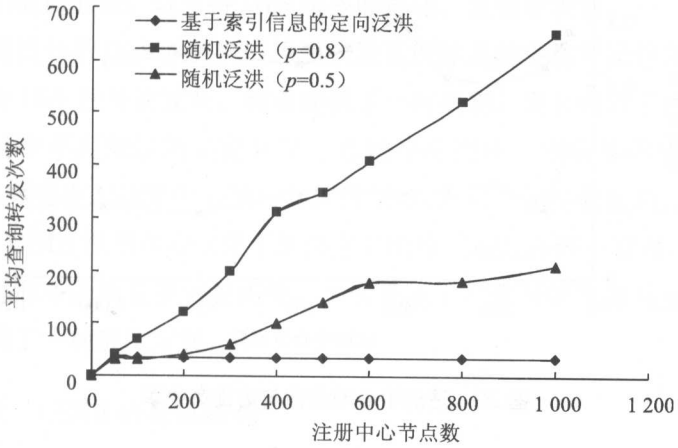

图 28　注册中心节点数与平均查询转发次数的关系

假设资源的元数据信息包含两个属性，查询成功表明在一次查询请求中查询条件与资源的两个相关属性相匹配。从图 29 可以看出，随机泛洪算法（p=0.5）的查询命中率最低，而基于索引信息的定向泛洪算法的查询命中率最高。

综合图 28 和图 29 的测试结果可以看出：①随机泛洪算法（p=0.5）由于在转发查询请求时没有依靠有用的信息进行转发，造成虽然转发查询的节点数较少，但是命中率非常低；②基于索引信息的定向泛洪算法由于使用了缓存信息和限定了查询范围，既通过减少

查询转发次数降低了消息负载，又提高了命中率，是非常有效的一种查询算法。

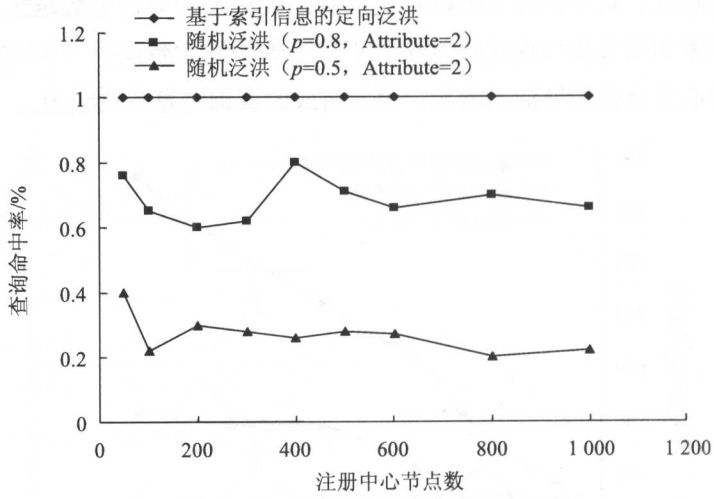

图 29　注册中心节点数与查询命中率

5.6　DSC 的设计与实现

前文讨论了数据服务中心的分布式存储模型，其中介绍了注册中心的组织，在这里给出注册中心的设计与实现。本小节首先分析了 UDDI 的数据结构和工作原理，然后讨论了基于 UDDI 的数据服务注册中心（Data Service Center，DSC）的设计，并根据实际应用需求，扩展了安全、管理等功能，给出了其参考实现。

5.6.1　通用描述发现集成协议

作为面向服务的体系结构的三个基本协议之一，UDDI 定义了一套标准的数据结构和编程接口，规定了面向服务的体系结构的三个主体（服务请求者、服务注册表和服务提供者）之间基于 SOAP 传输层的通信原理，以支持 Web 服务的描述、发布和查找。

通过分析 Web 服务的应用模式及其所涉及的主体和操作，UDDI 不仅为 Web 服务的发布、检索提供了一种机制，而且构成了动态服务调用中不可或缺的关键环节。在网络应用中，服务提供者遵循 UDDI 规范在 UDDI 中心注册其软件服务，而作为软件服务的元资源中心，UDDI 注册中心实现了软件服务的统一规划和统一管理，提高了软件服务的质量和可复用性，为实现基于互联网的资源共享与协作提供了一种解决方案。

5.6.1.1　UDDI 的数据结构

通过一组基于 Web 的访问协议及其实现标准，UDDI 主要用于解决当前基于互联网应用（如电子商务和电子政务）系统集成间的应用互通和互操作问题，它通过物理分布、逻辑集中的注册中心和对数据服务进行统一描述的 XML 应用程序来实现，其中，注册中心可以由很多提供 UDDI 注册服务的服务器所组成，成为一个集群；对于应用于局域网内的注册中心，尽管可能服务器数量不同，但是其数据结构及访问手段是一致的。

为了描述商业实体及其支持的 Web 服务信息，UDDI 协议定义了一套数据结构（见图 30），主要由商业实体信息（BusinessEntity）、服务信息（BusinessService）、绑定信息（BindingTemplate）、服务调

用规范的说明信息（tModel）以及发布者声明（PublisherAssertions）五种数据类型组成，其中，BusinessEntity 用于描述发布服务信息的商业实体的详细信息，BusinessService 用于描述商业实体所发布 Web 服务的信息，BindingTemplate 用于描述服务的访问点及构造规范的技术信息，tModel 用于描述服务或者分类法的规范（也是技术指纹的表现基础），PublisherAssertions 用于描述两个商业实体之间的关联关系，它们共同构成了 UDDI 描述服务的基础。

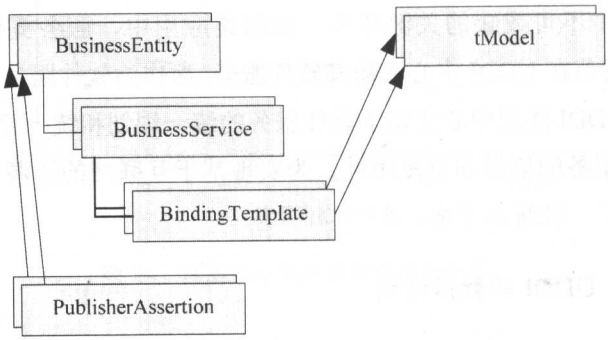

图 30　UDDI 数据结构

5.6.1.2　UDDI 的工作原理

从 UDDI 注册中心的角度来看，面向服务的体系结构中其他主体可以看作 UDDI 注册中心的客户端。作为面向服务的体系结构中的主体之一，UDDI 注册中心需要与其他主体之间进行通信，为此，除了提供一组标准的数据结构，UDDI 还提供了一种编程模型和模式，定义了与 UDDI 注册中心通信的规则。由于 UDDI 构建在基于 SOAP 的消息传输层之上，因此 UDDI 协议中所有的编程接口都用

XML 来定义，封装在 SOAP 信封中，在 HTTP 传输协议上传输。UDDI 的工作原理如图 31 所示。

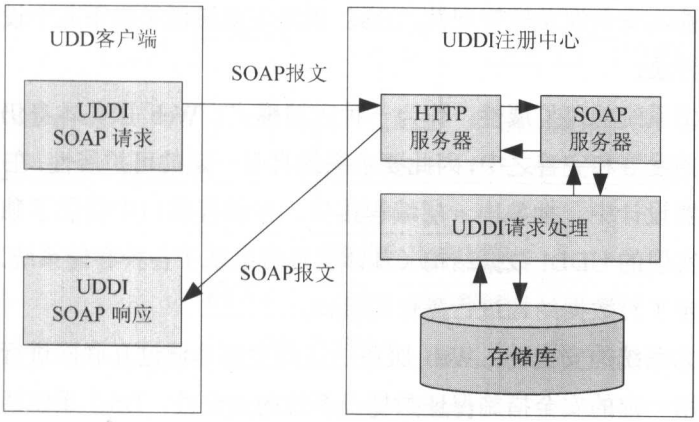

图 31　UDDI 的工作原理示意

首先，客户端的 SOAP 请求报文通过 HTTP 传输到 UDDI 注册中心；然后 UDDI 注册中心的 SOAP 服务器解析验证接收到的 SOAP 请求报文，并把 SOAP 响应报文返回给客户端。如果客户端发出发布、更新 Web 服务的 SOAP 报文，那么该 SOAP 报文首先要经过安全验证，以确保 UDDI 注册中心的安全性。

其次，基于面向服务的体系结构，设计并实现了基于 UDDI 的数据服务注册中心 DSC，主要由 DSC 客户端、DSC 服务器、DSC 管理控制台三个部分组成。下面将从系统的总体结构设计、系统各组成部分的实现机制和工作原理来进行介绍。

5.6.2 系统的总体结构设计

为了遵循 UDDI 协议，满足应用需求，保证系统具有一定的可扩展性、安全性及可管理性，DSC 系统主要遵循了以下几个设计原则和方法：

①系统的可扩展性：作为一种应用模式，Web 服务本身仍处于不断的发展和完善之中，因此要求系统具有一定的可扩展性。在 DSC 服务器设计中，抽象出一层编程接口，在编程接口中提供了独立于具体实现的 UDDI 数据结构（具体实现可以基于各种存储系统），并且规定了对数据结构进行操作的接口。

②系统的安全性：Web 服务的访问主要是通过互联网进行，要求采用一定的安全措施保证数据及系统的安全性，DSC 系统通过采用访问控制、用户授权、数据审计等多种手段来加强系统的安全性。

③系统的可管理性：根据 UDDI 注册中心的用途及范围，UDDI 注册中心可以分为：公共 UDDI 注册中心及私有 UDDI 注册中心，其中公共注册中心主要应用于互联网上，为全球的用户提供 Web 服务的访问服务；私有 UDDI 注册中心主要面向局域网以及企业内部。这两类注册中心都遵循 UDDI 协议，具有一致的数据结构和访问接口，但是由于面向领域的需求不同，私有 UDDI 注册中心需要满足一些特定的需求（例如定制用户类别），为此 DSC 提供了用户管理、策略管理、配置管理等功能。

根据上述的设计原则和方法，本书提出了 DSC 的总体结构（见图 32），它包括 DSC 客户端、DSC 服务器和 DSC 管理控制台等三部分，其主要工作原理如下：

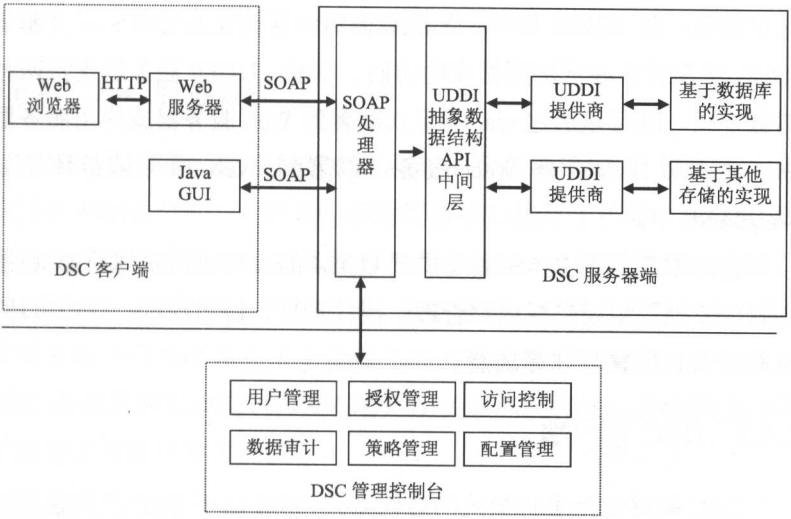

图 32　DSC 的总体结构

①DSC 客户端：其实现主要基于 UDDI4J 以及 Java XML 消息传输编程接口（Java API for XML Messaging，JAXM），并支持两种访问方式，一种是基于用户界面的客户端，通过 SOAP 协议与 DSC 服务器进行交互；另一种是基于浏览器的客户端，通过 HTTP 协议与 Web 服务器交互，然后再通过 SOAP 协议与 DSC 服务器进行交互。这两种客户端都支持发布、查找和更新 Web 服务。

②DSC 服务器：主要由 SOAP 处理器、UDDI 抽象数据结构 API 中间层以及基于不同存储设备的实现组成，其中 SOAP 处理器用于解析接收到的 SOAP 报文，并验证报文的有效性；UDDI 抽象数据结构 API 中间层独立于底层的实现，其优点是访问的透明性和扩展性，即不同的底层实现只要满足 UDDI 编程接口定义的接口，就能够向其上层提供一致的操作的接口。本书采用数据库构建 UDDI 的

底层实现，将 UDDI 规范定义的数据结构存储在数据库中，并基于 JDBC 实现对多种常用数据库的访问。目前，UDDI 服务器支持两种部署方式，既可以通过.war 的形式部署到 Web 服务器或应用服务器中，也可以作为一个 Web 服务，部署到 Web 服务运行环境如 WebSASE 中。

③DSC 管理控制台：通过扩展 UDDI 的编程接口实现对 DSC 服务器的管理，主要包括用户管理、授权管理、访问控制、数据审计、策略管理、配置管理等功能。

5.6.3 DSC 客户端

DSC 能够支持多种形式的访问方式，这是由于 DSC 的构建基于 SOAP 的消息传输层，因此无论客户端采用何种访问方式，只要注册中心能够接收并正确解析 SOAP 报文，就能够实现对 DSC 的访问。目前提供了两种形式的访问方式：

①基于 HTTP 协议的 Web 浏览器的 DSC 客户端。

②基于 SOAP 协议的 Java GUI 的 DSC 客户端。

SOAP 是一种在无中心分布式环境下用来交换结构化信息的轻量级协议，它描述了如何根据绑定框架规定的规则绑定到 HTTP 协议之上，可以使用多种底层协议进行 SOAP 消息的交换，解决分布式应用开发中的互操作性问题，因此，上述两种形式的客户端都是通过 SOAP 协议实现对 DSC 服务器的访问；同时，SOAP 消息本身是一个特殊格式的 XML 文档，客户端访问服务器的主要工作是根据 UDDI 编程接口构建 SOAP 报文，因此 DSC 的客户端支持两种方式来构建 SOAP 报文：针对系统的基本功能（如发布、查找、更新 Web 服务），使用 UDDI4J 来构建 SOAP 报文；针对用户管理（如注册、

登录、修改等功能）使用 JAXM 来构建 SOAP 报文。

5.6.4　DSC 服务器

DSC 服务器实现了 UDDI2.0 规定的全部编程接口，并根据实际应用需求如安全性、可管理性等对编程接口进行了扩展，使 DSC 服务器不仅能够提供 Web 服务操作的基本功能，而且提供用户管理、授权管理、访问控制、数据审计、策略管理、配置管理等扩展功能。此外系统还提供了发布多个企业之间关联关系声明的支持。下面从 DSC 服务器的实现机制、发布者声明关联匹配算法以及编程接口的扩展等方面进行阐述。

5.6.4.1　DSC 服务器的实现机制

抽象工厂模式是一种对象创建型模式，其目的是提供一个能够创建一系列相关或者互相依赖对象的接口，而无须指定它们具体的类。抽象工厂模式分离了具体的类，使客户与类的实现相分离，从而可以通过抽象接口操纵类实例。DSC 服务器的设计采用抽象工厂模式，把操作接口和数据结构与实现相分离，支持不同的 UDDI 实现，增强了系统的可扩展性。DSC 服务器主要包括 SOAP 处理器、UDDI 服务生成器、UDDI 服务实例等，其工作原理如下：

①SOAP 处理器：负责解析、验证 DSC 客户端发送的 SOAP 请求报文，之后交给 UDDI 服务生成器。

②UDDI 服务生成器：读取 UDDI 服务配置信息，并根据配置信息创建底层实现的服务实例。

③UDDI 服务实例：根据解析的 SOAP 报文决定调用合适的编程接口（目前支持 UDDI 2.0 规范定义的全部编程接口），主要包括

查询编程接口、发布编程接口及管理编程接口。

④编程接口通过访问数据存储实现得到访问结果，然后把访问结果封装成 SOAP 报文，沿原路返回。

在 UDDI 的实际应用中，将会存在大量的查询请求，而为了保证系统的性能，DSC 服务器采用了连接缓冲池的机制来保证系统的性能。这是由于目前 UDDI 服务的数据存储是基于数据库来实现，而在数据库处理中，资源开销最大的是建立数据库连接。如果每一个用户访问时，都重新建立连接，不仅用户需要长时间等待，而且系统可能会由于资源消耗过大而使性能急剧下降。因此，采用连接缓冲池的机制，重用已建立的数据库连接，以提高整个系统的性能。

系统使用了缓冲池之后，当一个请求到来的时候，首先查看缓冲池中是否有未被使用的连接；如果有，就使用该连接，如果没有，则建立一个新连接；获取并显示数据之后，将连接归还缓冲池。

为了尽可能地最小化缓冲池中的空闲连接，系统维护每个连接的最近使用时间标记和正在使用标志。当第一次获得连接时，连接的最近使用时间标记被设置为当前时间，连接的正在使用标志则被设置为真。

为了确定哪个连接是空闲的，将检查连接使用标志和时间标记，这个操作是通过周期性地获取连接缓冲池信息来实现的：

①查看正在使用连接的最近使用时间标记。如果最近使用时间和当前时间之间的时间差大于最长周期，则本连接将被认为是一个残留连接，被归还给缓冲池以供其他请求使用，它的正在使用标志被设置为假，且最近使用时间标记被设置为当前时间。

②检查未被任何请求使用的连接。如果最近使用时间与当前时间的时间差超过了最长空闲时间，将认为本连接是空闲的，从缓冲

池中除去。

5.6.4.2　发布者声明关联匹配算法

在 DSC 中，注册服务的企业之间可能具有一定的关联关系，如合作伙伴关系、母子公司关系等。为了描述两个企业之间的关联关系，UDDI 定义了发布者声明（PublisherAssertions）数据结构，它可以由两个企业中的任一个进行发布，但只有在对方匹配之后，该声明的状态才成为完整的（Status：Complete），否则是不完整的（Status：InComplete）。本书设计了以下的发布者声明关联匹配算法来实现匹配一个发布者的声明关联策略：

①输入：

新发布的关联声明：NPA。

已发布关联声明列表：<PublisherAssertions>元素描述为 PA1，PA2，…，PAn。

状态列表：S=<S1，S2>，其中，S1= Complete，S2= InComplete。

②开始遍历 PAi 元素（$1 \leqslant i \leqslant n$），$i$ 从 1 循环到 n，读取每个<PublisherAssertion>元素。

③在{PA1，PA2…PAn}中匹配到与 NPA 关联声明中的两个企业都同名的元素 PAk。

```
if PAk 的 Status 属性为 S1   then
    break;    //不执行任何操作
    else   //对于 Status 属性不为 S1 的声明，可能会执行匹配操作
    if PAk 的发布者与 NPA 的发布者相同，then
        break; //不执行任何操作
else
```

进行匹配操作，该声明的状态设置为 S1。

④if NPA not in { PA1，PA2…PAn } then

if NPA 的发布者是该声明中的两个企业的所有者

已发布关联声明列表中增加一条关联声明，该声明的状态设置为 S1。

$n=n+1$;

else

已发布关联声明列表中增加一条关联声明，该声明的状态设置为 S2。

$n=n+1$;

⑤PAi 遍历结束。

⑥输出结果：{PAi|$1 \leqslant i \leqslant n$}

5.6.4.3 基于 UDDI 协议的编程接口的扩展

在 UDDI 编程接口规范中，所定义的编程接口能够实现基本的 Web 服务访问功能，但对于部署在不同的场景中供多个用户使用的注册中心来讲，这些还显得不够。它还需要考虑系统的安全性、策略管理等，而这些功能特性在 UDDI 协议中并没有进行定义，需要由系统的提供者来实现。因此，本书的 DSC 对 UDDI 编程接口进行了扩展，其扩展分为两个部分：

①原有功能编程接口的扩展（如查找服务编程接口）：UDDI 提供了两种查找服务的模式：一种是浏览模式，即服务请求者可以根据通用的分类标准来搜索，并逐步缩小查找的范围，直到找到满足需要的服务；另一种是直接获取模式，即通过唯一的关键字直接得到特定服务的描述信息。对于第一种方式，需要提供发布该服务的

商业实体键值，但在实际应用中，服务请求者往往需要能够仅根据 Web 服务的名字直接查找 Web 服务，并不一定知道企业键值，因此，原有的编程接口已经不能满足应用的需求，本书在原有编程接口的基础上进行了扩展，满足应用需要。

②新增功能编程接口的扩展：本书定义了一组编程接口，用于提供对这些新增功能（如用户管理、授权管理、访问控制、数据审计、策略管理以及配置管理等）的访问。

基于协议进行扩展，如果不作适当的处理，通常会影响到系统的通用性。为了在对协议进行扩展的同时，仍能保证系统的通用性，本书通过采用名域空间来区分协议原有的编程接口与扩展的编程接口。客户端如果使用扩展的编程接口访问 DSC 服务器，那么在构建 SOAP 报文的时候，需要 SOAP 报文的头部加上特定的名域空间；这样，服务器在解析 SOAP 报文的时候，根据名域空间就能够区分原有的编程接口和扩展的编程接口。与 CapeClear 相比较，具有更好的扩展性。

5.6.5　DSC 管理控制台

DSC 管理控制台是一个框架程序，它将对 DSC 的不同的管理功能集成到一个统一的管理界面上，从而方便对系统的管理。目前管理控制台所提供的功能包括：用户管理、授权管理、访问控制、数据审计、策略管理以及配置管理等。

访问控制机制中，有两种控制粒度，一是基于网络连接的粗粒度控制，二是基于具体应用对象的细粒度控制。在用户管理功能中，采用了细粒度的访问控制技术，并结合身份认证，通过细粒度访问控制来加强对 UDDI 服务的控制，进而使不同身份的用户（操作站

点管理员、服务发布人员、普通的访问人员等）对资源具有不同的访问权限和服务操作。

注册在DSC的服务信息，普通用户只可以执行访问，而对于更新Web服务信息等操作，则需要发布者本人才能进行操作。但是这种机制仍然不能满足企业内部的私有UDDI注册中心的需要，因为即使对有些信息执行访问，也只有经过发布者授权，其他用户才能够访问。为了保证在DSC注册的Web服务信息的机密性，UDDI管理控制台提供了授权管理功能，使发布者在发布Web服务时，可根据需要选择用户或者用户组，并赋予他们不同的操作权限。为此，定义了一个授权的数据结构：authorityBag（见图33）。

```
<element name = "authorityBag">
 <complexType>
  <sequence>
   <element ref = "aclPair" minOccurs = "0" maxOccurs = "unbounded"/>
  </sequence>
 </complexType>
</element>
<element name = "aclPair">
  <complexType>
   <attribute name="op"  type="string" use="requried"/>
   <attribute name="user" type="string" use="required"/>
   <attribute name="userType" type="string" use="required"/>
  </complexType>
</element>
```

图33　authorityBag数据结构

authorityBag引用了0到多个aclPair元素。aclPair元素具有三个属性，分别是：op，user，userType。其中op定义了操作类型，user是访问人员，可以是一个用户，也可以是一个用户组，由userType属性来加以区分。在UDDI的五个基本数据结构如BusinessEntity、

BusinessService、BindingTemplate、tModel 中加入 authorityBag 这样的一个元素，用于描述对该数据结构操作的授权。

数据审计用于记录访问者、资源以及该资源被访问的时间。因为每一个用户都需要认证之后才能访问受保护的资源，加强了用户的责任意识，并且每一个访问试图都能够被记录下来，从而增强了系统的安全性。此外管理控制台还提供了策略管理和配置管理，用于对用户发布 Web 服务数目的策略以及数据源进行管理。

5.7　本章小结

本章深入研究了数据服务领域中面临的关键问题，并提出了相应的解决方案。首先分析了信息共享中所面临的关键问题，提出了一种分布式信息存储模型，用于资源的发布和发现，提高了可扩展性和灵活性。针对该模型，提出了几个关键技术作为数据服务中心效率、准确度和有效性等问题的解决方案：注册中心的组织、信息动态注册与更新机制、资源发现，并且设计实现了基于 UDDI 的数据服务注册中心。

第 6 章
数据交换平台的设计与实现

数据交换平台作为一个面向分布式应用的消息中间件，通过建立统一的信息交换模式，能够对不同的应用系统中的数据进行交换和处理，消除了各应用之间的协议差异、平台差异和数据差异。随着电子政务和电子商务的进一步发展，数据交换平台也必将发挥更大的作用。本章在前面几章对基于 Web 服务的数据交换系统框架的一些关键技术分析的基础之上，提出了一种新的基于覆盖网络的数据交换协议，描述了协议的框架和构成，设计实现了基于 Web 服务的数据交换平台，讨论了其各个组成部分的工作原理和实现机制，并讨论了数据交换平台在电子政务中的应用部署。

6.1 引言

电子政务建设的核心问题是实现各个政府部门之间的互联互通及互操作，其实质是实现信息的共享和交换。同时，信息共享与交换也是广域网络下信息系统的核心问题之一。由于广域网络的复杂性和电子政务应用系统的异构性，使得信息交换很难在 OSI（Open System Interconnection，开放式系统互联）模型的应用层以下实现。

近年的研究通常把信息系统构建在特定的覆盖网络上，利用覆盖网络屏蔽物理网络的多样性，实现不同的应用目标和服务质量。

设计一个基于覆盖网络的数据交换协议，在覆盖网络上实现信息交换，将带来一些便利，同时也面临着网络不可靠、不同的底层传输协议、消息寻址和路由转发、端到端的安全等问题。此外，支持大数据量的传输、可靠传输、断点续传等能力是协议及其实现应该具备的另一些重要能力。

目前有不少数据传输和数据交换协议可以在覆盖网络上使用，如 SOAP、ebXML MS 等。SOAP 提供了基于 XML 的消息定义，可以用于在分布式环境中的节点之间交换结构化的有类型的信息。但是 SOAP 的使用限于短消息而不是大小不限的消息，支持几种特定的消息类型而不是实现通用的处理。ebXML MS 定义了特定的报文结构以支持业务信息可靠、安全地传输，但由于扩展了 SOAP，其报文大小也受到一定的限制。

目前互联网上兴起的即时通信软件实现了各自的消息机制，可以用来传送短消息、文件等数据信息，然而它们主要应用于人与人之间的交流，对安全性的要求较弱，且这些软件都针对特定的应用模式。它们的数据都是由发送者“推”给接收者，接收者不能主动去“拉”数据；而基于 Gnutella、Pastry、Chord、Freenet 等实现的 P2P 软件通常由请求方检索和获取文件，将它们作为通用的数据交换协议存在一定的困难。

本章提出一种新的基于覆盖网络的数据交换协议（Data Exchange Protocol，WSDEP），它是建立在物理网络之上实现各个应用之间数据传输的一种软通道，能够传输结构化和非结构化的数据，支持可靠传输、断点续传和端到端的安全，并且适用于不同的覆盖

网络，在WSDEP的基础之上实现了WS-XP（data eXchange Platform，基于Web服务的数据交换平台）。

6.2 数据交换协议

本节首先分析了基于覆盖网络的数据交换协议所面临的问题，它们应能够提供什么样的服务并具有何种能力，然后描述了WSDEP的框架和构成。

6.2.1 问题分析

数据交换是双向的，系统A既可能向系统B请求数据，也可能向系统B提供数据。然而由于网络地址转换技术和防火墙技术的广泛使用，使得某些主机可能无法接收到来自其他互联网的主机的连接请求。因此，直接在IP层构造一个通用的数据交换协议是不可行的，这将导致某些主机无法参与到信息交换中。一个可行的解决办法是在实际的物理网络之上再构造一个逻辑网络，称为覆盖网络。把信息交换系统构建在覆盖网络上，在应用层实现信息交换，解决了网络可达性的问题，同时屏蔽了真实网络的多样性。

在网络中，数据流动的方向是任意的，任意节点都可能同时提供数据和消费数据。数据交换协议向上层应用提供两个服务，一个是推数据服务，数据提供者主动向数据消费者发出请求，并在获得提供者确认后，提供者再将数据传输给消费者；另一个是拉数据服务，由数据消费者向数据提供者发出请求索要数据，并在获得提供者确认后，等待数据提供者将数据传输给自己。利用这两个服务，可以直接实现任意两个节点之间双向的数据交换。通过组合这两个基本服务，可以实现更为复杂的数据交换模式或应用。

作为一个通用的、基于覆盖网络的数据交换协议，应该具有传输多种数据类型、断点续传、端到端的安全、可靠传输等功能。

6.2.2　假设和说明

WSDEP 的目的是在覆盖网络上实现安全、可靠的信息交换，提供推数据和拉数据服务，支持不同的数据类型，具有可靠传输、断点续传和多传输协议绑定的能力。协议被定义为以全双工方式进行信息交换，并且信息交换模式是异步的。协议假设传输信道是不可靠的，可能会引起任意的消息失真、丢失、重复、重排等情况。协议需要向上层应用提供有确认有连接的服务。

6.2.3　词汇和消息

协议定义了 7 种消息，分别是拉数据请求、拉数据响应、推数据请求、推数据响应、传输数据请求、传输数据确认和故障消息。图 34 是对消息结构的说明，其中符号‘|’表示选择关系，符号‘？’表示元素出现 0 次或者 1 次。

其中：

①TransactionID 是一次信息交换会话的标识。

②MessageID 是消息的序号。

③Routing 记录路由信息，From 和 To 分别表示消息的发起结点和消息的接收结点。Hops 表示了消息在覆盖网络上的最大跳数。

④PullRequest，PullResponse，PushRequest，PushResponse，TransportRequest，TransportResponse 和 Fault 对应了七种消息。

⑤TimeStamp 表示时间戳信息，可选。

⑥Features 表示协议支持的一些附加特性。IsSecurity 表示是否

采用安全传输。

```
<DEPMessage TransactionID="(ID)" MessageID="ID">
    <Routing>
     <From></From>
     <To></To>
     <Hops/>
    </Routing>
          <PullRequest/> | <PullResponse/>  |  < PushRequest/> | <
     PushResponse/> | < TransportRequest/>  | < TransportResponse/> |
     <Fault/>
    <TimeStamp/>?
    <Features>
     <IsSecurity>yes | no</IsSecurity>
    </Features>
    </DEPMessage>
```

图 34　WSDEP 消息结构

6.2.4　原语和过程规则

协议定义了两个服务原语：pushData 和 pullData，分别实现了一次完整的推和拉数据。

每个服务原语的实现是多个消息的交互过程，其中使用了三次握手机制。以 pullData 为例，具体的过程是：

①第一次握手：发送方向接收方发送连接请求，请求内容主要是数据描述，除此之外还包括需要收发双方协商的传输配置，如是否传输安全、数据切片大小等信息。如果要求安全特性的支持，那么请求内容包括发送方的安全证书和防止重放攻击的随机数，请求内容用接收方的公钥加密。

②第二次握手：接收方接收到发送方的连接请求后，向发送方发送连接响应，响应内容用发送方的公钥加密。

③第三次握手：发送方收到接收方的连接响应后，再将接收方生成的随机数和对称传输密钥返还给接收方，验证双方的身份。

当收发双方通过三次握手建立起端到端的安全数据传输通道后，双方开始利用该通道传输数据。数据传输过程中，将对大数据进行分片，并采用滑动窗口协议传输数据。由于网络固有的不可靠因素，会导致分片的丢失。对于这种情况，由滑动窗口协议保证传输的可靠性。

6.3　WS-XP 的设计和实现

基于 WSDEP，设计并实现了 WS-XP，它主要由数据交换引擎、适配器模块、传输模块等组成。目前，WS-XP 已经在一些政府机构中得到了应用，下面将从系统的总体结构设计、系统各主要组成部分的实现机制和工作原理来进行介绍。

6.3.1　系统的总体结构设计

为满足应用需求，WS-XP 主要遵循了以下几个设计原则和方法：①可扩展性：由于数据类型的多样性，数据交换和数据描述必须以 XML 为基础，从而能够支持异构数据的交换；②可靠性：提供断点续传功能，保证系统的可靠性；③传输效率：为了实现穿越防火墙、跨平台的数据交换，使用 SOAP 作为传输协议，但由于 SOAP 自身的特点，需要采取一定的措施来保证系统的传输效率。

根据上述的设计原则和方法，提出了 WS-XP 的体系结构（见图 35），它包括数据交换引擎、传输模块、适配器模块等。

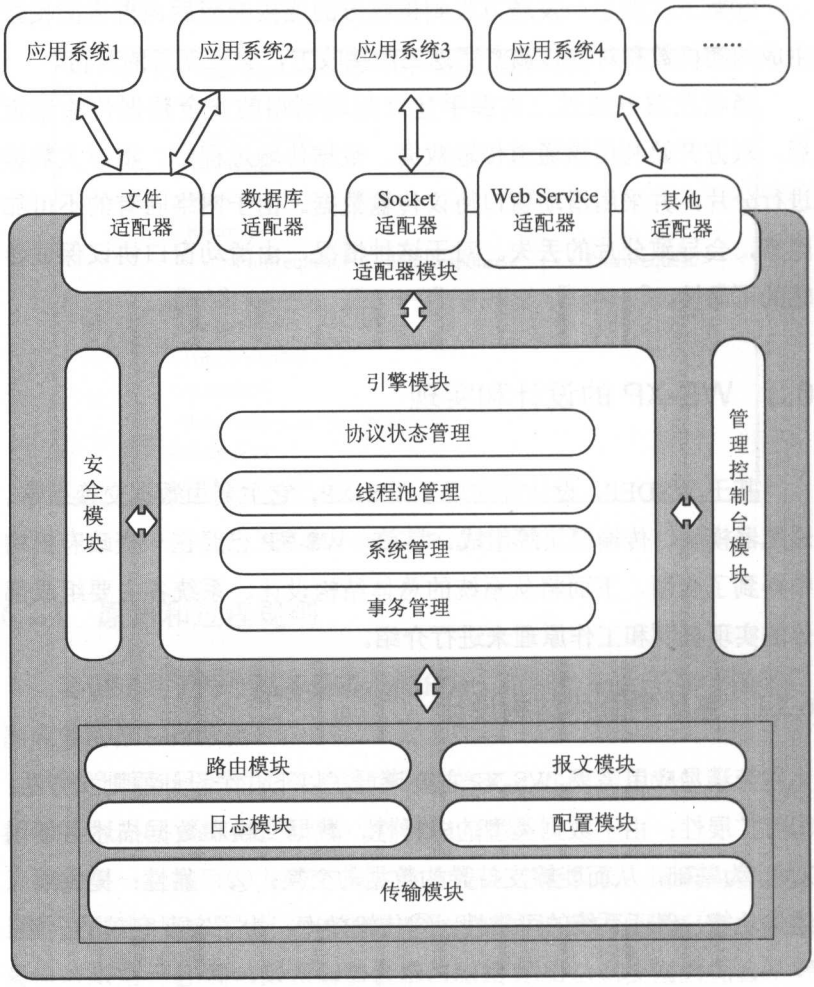

图 35 WS-XP 体系结构

①数据交换引擎：是 WS-XP 的核心模块，负责系统的启动和停止、对整个数据交换流程进行控制，以及为管理控制台提供运行状态监控

接口，在数据传输过程中，引擎还负责调度 WS-XP 的其他子模块。

②适配器模块：应用系统要通过 WS-XP 与其他的应用系统进行数据交换，就必须通过 WS-XP 提供的接入方式与 WS-XP 进行数据交互。在 WS-XP 中，这种接入方式是通过适配器机制来提供。

③传输模块：提供数据发送和数据接收功能，支持不同的传输协议。

此外，安全模块用于提供数据内容的安全机制、端到端的传输安全机制、身份认证和授权管理机制；路由模块维护路由表，并对消息中的路由元素项进行解析，提供数据路由服务；管理控制台模块、配置模块和日志模块用于对系统进行配置和管理；报文模块定义数据交换系统所使用报文的结构，实现一套针对该报文结构的 Java 对象描述 API，提供给程序员用于进行数据交换系统报文的生成、修改和信息提取等操作。

6.3.2　数据交换引擎

数据交换引擎是 WS-XP 调度和启动的核心，它负责系统的启动和停止、对整个数据交换流程进行控制，以及为管理控制台提供运行状态监控接口，在数据传输过程中，引擎还负责调度 WS-XP 的其他子模块。下面将针对数据交换引擎的主要结构、引擎模块之间的关系以及引擎的三个主要流程进行分析。

6.3.2.1　数据交换引擎的结构

数据交换引擎是 WS-XP 的核心模块，也是结构最为复杂的模块，图 36 说明了引擎模块的内部结构、与外部模块关系以及在系统中的位置。

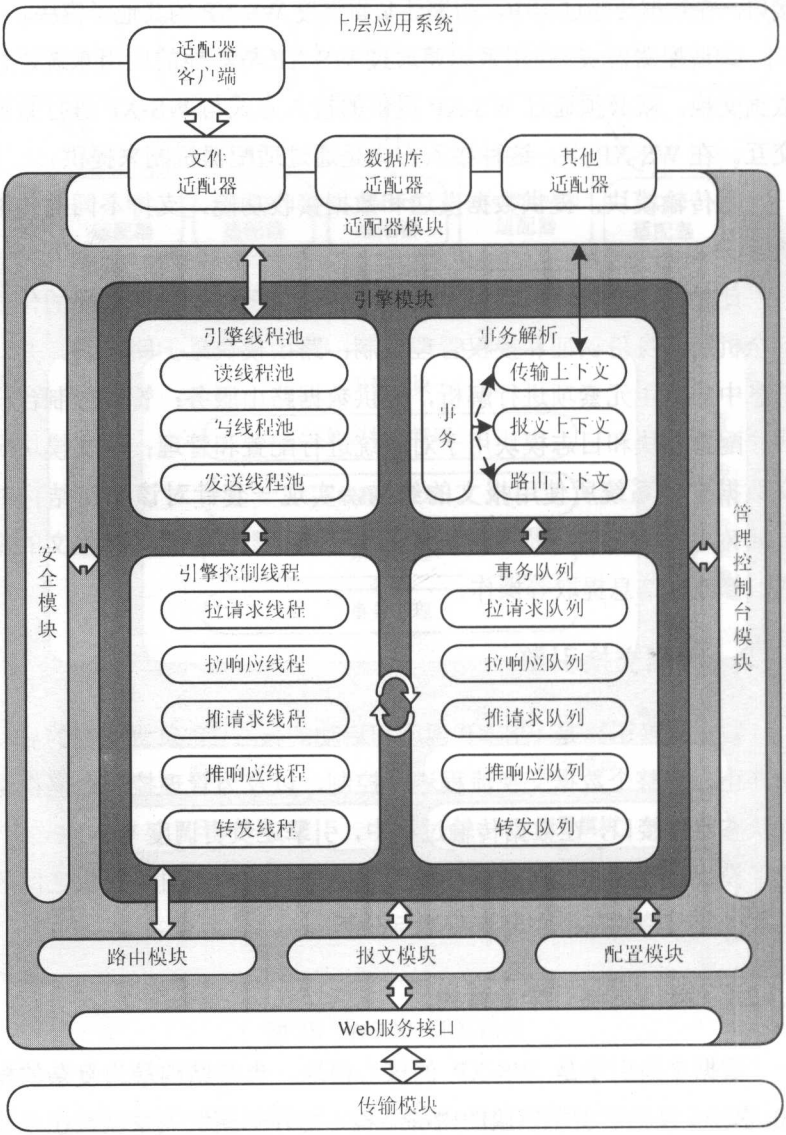

图 36　数据交换引擎内部结构

引擎模块各个类之间存在复杂的引用关系，用 UML 类图的方式描述了程序之间的继承关系，如图 37 所示。

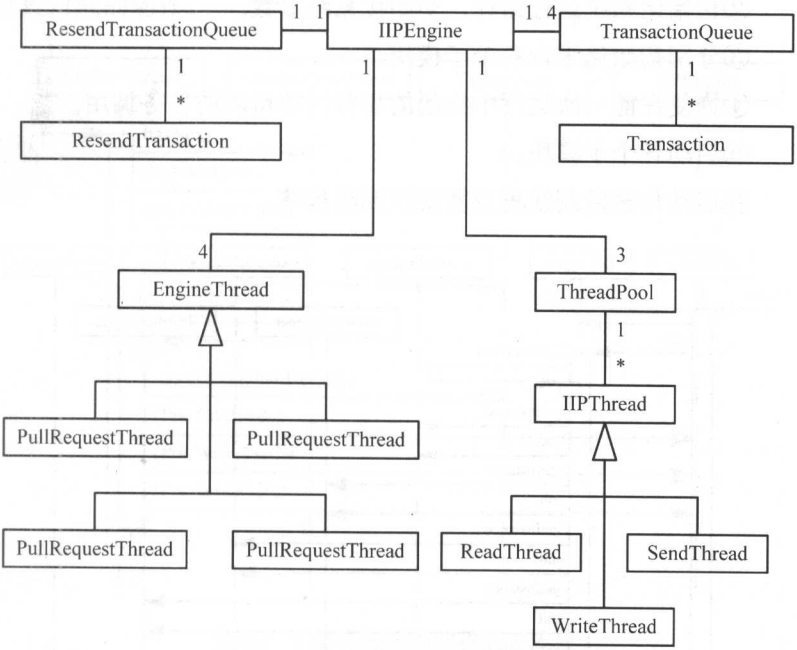

图 37　数据交换引擎程序继承关系

6.3.2.2　数据交换引擎的主要流程

数据交换引擎模块程序之间的调用主要发生在三个运行流程中：系统启动流程、传输控制线程处理流程和接收报文处理流程。下面用 UML 顺序图的方式描述三个运行流程的程序调用关系。

（1）系统初始化流程

WS-XP 的启动及初始化过程由引擎模块负责，它的启动过程如

下（见图 38）：

①载入平台的配置信息。

②初始化系统运行路径，初始化系统参数。

③分别初始化平台各个子模块。

④恢复在前一次运行中挂起的事务，等待新的任务调用。

⑤启动各个子模块。

⑥启动传输控制线程及转发控制线程等。

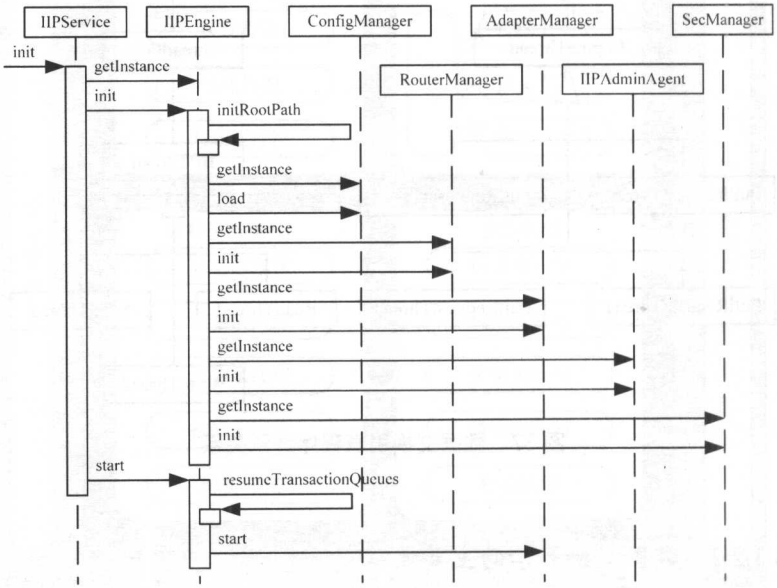

图 38　系统初始化流程

（2）引擎传输控制线程处理流程

数据交换是 WS-XP 的核心功能，其流程由数据交换引擎控制，具体处理过程如下（见图 39）：

①从事务队列中获取事务。

②根据事务状态及传输状态自动机进行状态跳转，并执行跳转过程对应的动作。

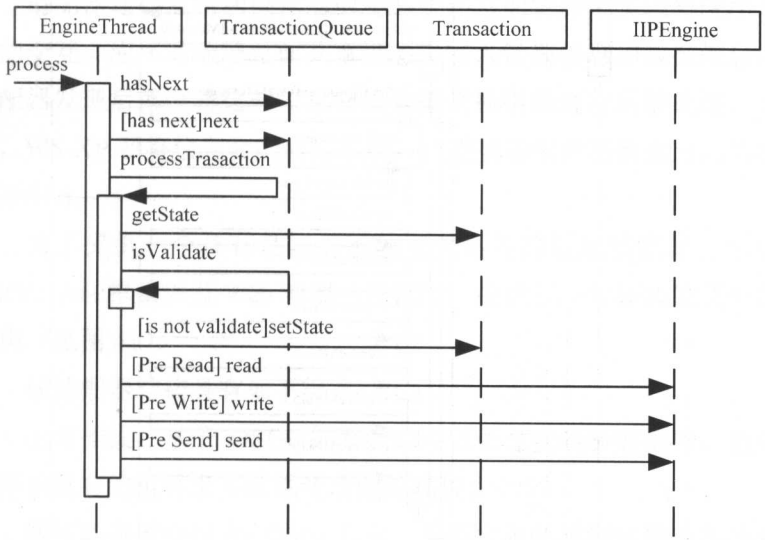

图 39　引擎传输控制线程处理流程

（3）报文接收处理流程

WS-XP 传输数据是通过 Web 服务调用的方式实现的，当 Web 服务接口接收到一个传输报文后交由引擎进行处理，具体处理过程如下（见图 40）：

①解析传输报文。

②根据报文获取或构造相应事务。

③修改事务状态。

④存储接收到的附件到数据包装类中。

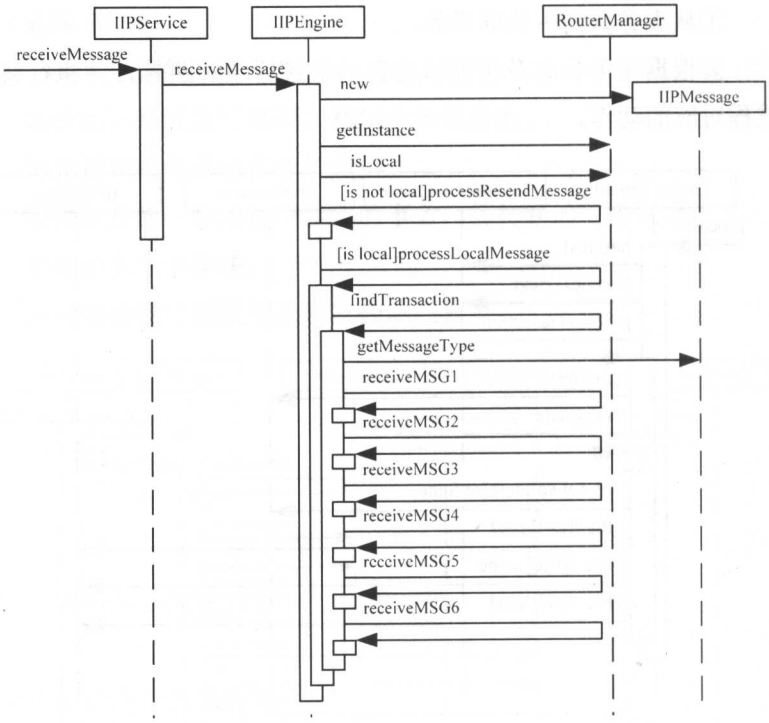

图 40　报文接收处理流程

6.3.3　适配器机制

适配器技术从广义上可以理解成为一种通过互联网进行数据及功能共享的技术，狭义上可以将其理解成为一种介于上层应用和数据交换平台之间的接口。在电子商务、电子政务应用领域，适配器技术已获得了大量的应用。它是一种信息集成的手段，通过特定的技术标准访问服务，为应用间的交互提供一致的访问接口。

应用系统主要通过适配器来使用 WS-XP，即应用系统把数据交

给适配器，再由适配器通过数据交换引擎把数据发送出去；或者数据交换引擎将接收到的数据通过适配器传递给应用系统。应用系统也可以直接调用引擎的接口发送数据。

适配器通过监听和操作与应用系统共享的资源来实现与应用系统的交互。应用系统通过向共享资源中写入管理信息和数据信息，适配器发现后就会将这些信息交给数据交换引擎进行后继处理。同样，WS-XP 要将信息交给应用系统，也需要适配器将数据写入共享资源中。

为了提供一个具有良好扩展性及灵活性的适配器框架，结合 OSGi、Ant 框架以及 Web 容器三种框架，提供了一种新型的适配器结构（见图 41）。

该适配器结构具有如下特点：

①可插拔：基于 OSGi 的系统，可以通过安装新的组件、更新或停止现有的组件来实现系统功能的插拔。

②可动态改变行为：OSGi 具有一整套完整的机制实现动态改变系统行为，可插拔和可动态改变行为从根本上保证了系统具有足够的灵活性和扩展性。

③稳定高效：基于 OSGi 的系统采用了微核机制，微核机制能够保证系统的稳定性，基于微核机制的系统只要微核是稳定运行的，那么系统就不会受到其中的组件的影响。

WS-XP 适配器提供的主要功能包括：

①提供多种数据接入方式：考虑到应用种类繁多，编程语言也各不相同等情况，为了使系统能够具有更好的适用性，需要提供多种方式以及可扩展的结构，从而为多种数据接入方式提供一个统一的运行环境。

②提供可扩展的适配器框架：为了能够快速地添加新增功能并去掉无用功能，在不影响其他的模块的情况下，提供一个扩展性强、模块间耦合度较低的适配器框架。

③提供数据读取和预处理功能：数据处理类型包括文件、数据库、流，它的功能包括读、写数据以及文件定位、切片和打包等。

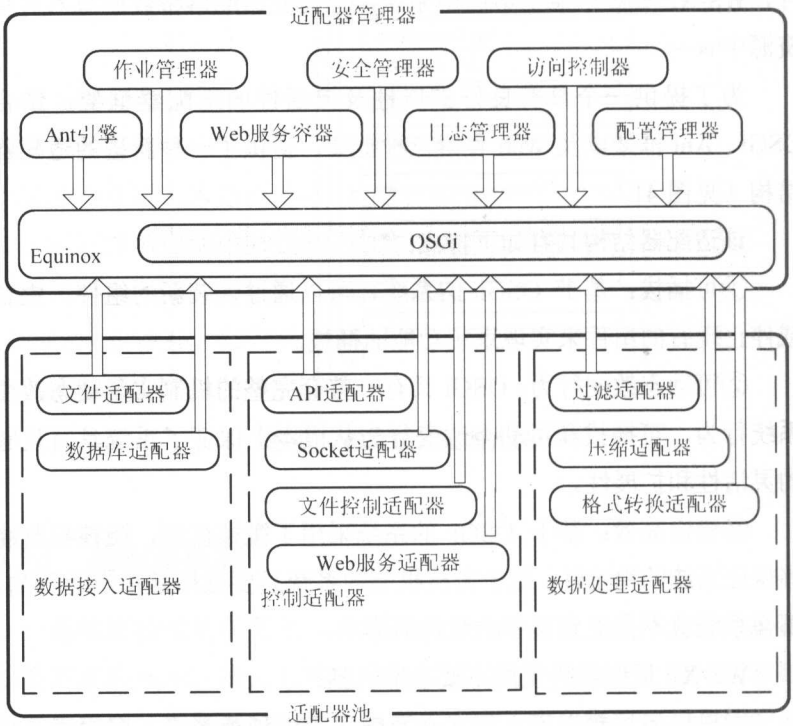

图 41　适配器结构

6.3.4　传输模块

传输模块是系统核心模块，能够接收和响应对 Web 服务的访问请求，负责管理和控制其他子系统和模块。传输模块主要任务包括：

①管理消息发送和接收队列。

②注册和注销任务线程。

③调用其他子系统进行消息传输。

④在接收消息时，调用回调函数通知任务线程。

⑤提供 SOAP 绑定功能，实现 SOAP 和多种传输协议的绑定。

传输模块采用消息队列技术对请求消息和响应消息进行管理。消息队列技术是一种分布式应用间交换信息的技术，消息队列可以存储于内存或者磁盘介质上，直到被相应的线程或应用程序读走。通过使用消息队列，应用程序可以独立地执行，它们不需要知道彼此的位置，或者在继续执行前不需要等待接收线程接收此消息。通过对消息队列的序列化/反序列化，可以确保消息没有丢失，从而能够保证消息传输的可靠性。

6.3.5　事务机制

在 WS-XP 中，一次完整的数据交换被称为一个事务。事务对象中完整地保存了一次信息交换的当前状态和上下文信息。数据交换引擎为事务建立队列，以实现对并发事务的控制和维护。数据交换引擎利用事务的持久化机制实现了故障恢复和断点续传。

在事务机制中，为了支持故障恢复，需要解决如下几个问题：

①何时进行序列化：对事务队列进行修改操作，或者对事务队列中的事务进行修改操作的时候，就需要保存到外存中，从而能够

最大限度地保存数据。也就是需要考虑保存的频率问题，它会对性能产生重大影响。

②何时进行恢复：在系统启动的时候，由数据交换引擎进行恢复。

③序列化的存储介质：目前系统采用 XML 文件。针对每个事务建立一个文件，以事务 ID 作为文件的名称。针对四种不同类型的事务队列，建立四个文件夹，每个文件夹下面是一个事务队列，系统只需要提供故障恢复文件的根路径即可。

④哪些内容需要进行序列化：事务队列中所有的事务，以及与事务相关的关键信息需要进行序列化。

在解决了上述问题之后，提出了故障恢复的具体方法：

①在 WS-XP 的处理流程中，定义一系列的故障恢复点。

②当流程执行到这些故障恢复点时，持久化当前事务信息。

③系统发生故障则流程回滚到最近的故障恢复点，一旦系统重新启动之后，便从持久化的事务信息中读取出相应的恢复点信息，继续故障前的执行流程完成数据交换。

6.4 WS-XP 的应用部署

针对大规模数据交换所面临的应用系统的异构性、数据类型的多样性、数据传输的可靠性以及数据传输的效率等问题，在上述设计内容的基础上，设计实现了 WS-XP，并且把它应用于某省电子政务平台系统。

某省电子政务平台系统是基于数据交换平台 WS-XP 研制开发的一个支持政务协作的信息交换平台，它的目标是能够实现全省范

围内政府部门之间的信息交换，支持政府部门之间的并联审批和协同办公。该平台已完成在省政府办公厅（中心平台）、省级重点部门（厅局节点）以及部分地市级单位的部署，数据交换的范围覆盖了整个省政府，实现了省政府内横向和纵向的数据交互。某省电子政务平台系统的应用部署情况如图 42 所示。

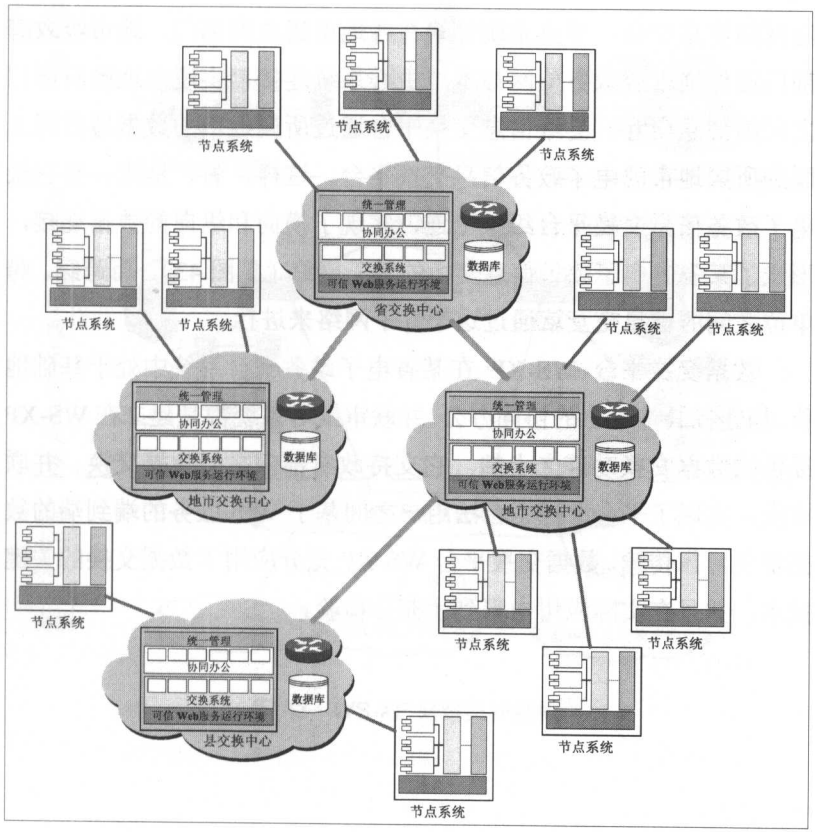

图 42　WS-XP 应用部署结构

在某省电子政务平台系统中，在省、市、县各级中心设立了信息交换中心系统，在各个委办局设立了信息交换节点系统，其中省级信息交换平台的中心系统部署在省政府办公厅信息中心，节点系统部署在各省级政府部门，省办公厅和其他省级政府部门通过政务外网与省信息交换平台中心系统连接，实现省级政府部门之间的信息交换和协作办公。地市级信息交换平台的中心系统部署在地市政府网络信息中心，节点系统部署在各地市级政府部门，地市级政府部门通过地市级政务外网与本地中心系统连接，实现本地政府部门之间的信息交互。县级信息交换平台通过所属地市的数据通信网上联到所属地市的电子政务信息交换平台。这样，省、地市、县三级电子政务信息交换平台级级上连，实现了横向和纵向的数据连接，形成了某省省电子政务信息交换的骨干网络（见图 43），而跨级、跨单位之间的信息交互就通过这个骨干网络来进行。

数据交换平台 WS-XP 在某省电子政务平台系统中处于基础地位，构建在其基础上的协同办公、并联审批等系统都是建立在 WS-XP 提供的数据交换功能之上的。它支持政府部门之间数据交换、并联审批，实现了平台上任意合法用户之间基于 Web 服务的端到端的数据安全可靠传输。数据交换平台 WS-XP 充分应用了数据交换的关键技术，并且在实际应用中得到了充分检验。

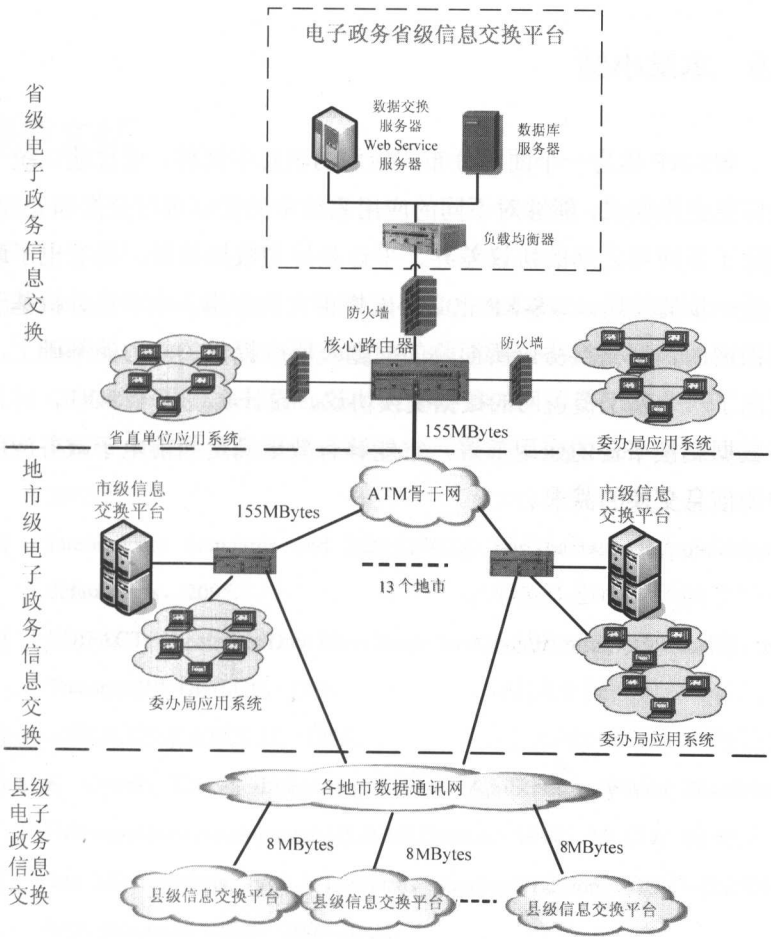

图 43　WS-XP 网络拓扑结构

6.5 本章小结

WS-XP 作为一个面向分布式应用的消息中间件，通过建立统一的信息交换模式，能够对不同的应用系统中的数据进行交换和处理，消除了各应用之间的协议差异、平台差异和数据差异。随着电子政务进一步的发展，WS-XP 也必将发挥更大的作用。本章在分析基于覆盖网的数据交换协议所面临的问题及所应提供的能力的基础上，提出了一个基于覆盖网的数据交换协议，设计实现了 WS-XP，讨论了数据交换平台的应用部署，它能够有效地满足当前电子政务应用中的信息交换的需求。

参考文献

[1] Adam N R, Adiwijaya I, Atluri V, et al. EDI through a distributed information systems approach[A]//Proceedings of the Thirty-First Hawaii International Conference on System Sciences[C]. USA, 1998.

[2] What's edi[EB/OL]. http://www.portinfo.net.cn/edispec/edikwg/edi3.php, 2002.

[3] International Standards and EDI[EB/OL]. http://www.itu.int/publications/default.aspx, 2005.

[4] EDIFACT: Electronic Data Interchange for Administration, Commerce, and Transport[S]. UN/ECE, 1986.

[5] ANSIX.12[S]. AXCS.12, 1985.

[6] S. Vinoski. Corba: Integrating Diverse Applications Within Distributed Heterogeneous Environments[J]. IEEE Comm., 1997, 35 (2): 46-55.

[7] Sun Microsystems. Java 2 Platform, Standard Edition (J2SE) [EB/OL]. http://java.sun.com/j2se. 2006-10-22.

[8] Davis A, Du Zhang. A comparative study of DCOM and SOAP[A]//Fourth International Symposium on Multimedia Software Engineering[C]. 2002: 48-55.

[9] Martin Gudgin, Marc Hadley et al. SOAP Version 1.2 Part 2: Adjuncts, W3C Recommendation[EB/OL]. W3C Group. http://www.w3.org/TR/2003/REC-

soap12-part2-20030624，2003.

[10] 柴晓路，梁宇奇. Web Service 技术/架构和应用[M]. 北京：电子工业出版社，2003：4-6.

[11] Martin Gudgin，Marc Hadley et al. SOAP Version 1.2 Part 2：Adjuncts，W3C Recommendation[EB/OL]. W3C Group. http://www.w3.org/TR/2003/REC-soap12-part2-20030624. 2003-06-24.

[12] Erik Christensen，Francisco Curbera et al. Web Services Description Language（WSDL）1.1 W3C Note[EB/OL]. W3C Group，15 March 2001. http://www.w3.org/TR/wsdl.html.

[13] Universal Description，Discovery and Integration（UDDI）Protocol[EB/OL]. http://www.uddi.org，2002.

[14] Hsiu-Hui Lee，Chun-Hsiung Tseng.A software framework for Java message service based Internet messaging system[A]//Proceedings of the Fourth International Conference on Parallel and Distributed Computing，Applications and Technologies（PDCAT'2003）[C].2003：161-165.

[15] Chand R，Felber PA. A scalable protocol for content-based routing in overlay networks[A]. Proceedings of the 2nd IEEE Int'l Symp. on Network Computing and Applications[C]. Cambridge，2003：123-130.

[16] Werner C，Buschmann C，Fischer S. Compressing SOAP messages by using differential encoding[A]. In：Proceedings of the IEEE International Conference on Web Services（ICWS'04）[C]. IEEE，2004：540-547.

[17] Paul V Biron，Kaiser Permanente. XML Schema Part 2：Datatypes Second Edition W3C Recommendation[EB/OL]. W3C Group. http://www.w3.org/TR/2004/REC-xmlschema-2-20041028. 2004.

[18] Zhuzhong Qian，SangLu Lu，Li Xie.Colored Petri Net Based Automatic

Service Composition[A]//The 2nd IEEE Asia-Pacific Service Computing Conference[C]. Tsukuba Science City，2007：431-438.

[19] Wenya Yang，Shaohua Tang. A Solution for Web Services Transaction[A]// International Conference on Hybrid Information Technology（ICHIT'06）[C]. Cheju Island，2006：416-423.

[20] Shuping Ran. A model for web services discovery with QoS[A]//ACM SIGecom Exchanges[C]. 2003.

[21] Vinoski S. Where is is middleware[J]. Internet Computing，IEEE，2002，6（2）：83-85.

[22] What's middleware[EB/OL]. http://www.webspherechina.net/club/viewthread.php？tid=9108，2007.

[23] Menasce D A. MOM vs. RPC：communication models for distributed applications[J]. Internet Computing，IEEE，2005，9（2）：90-03.

[24] WebSphere Information Integrator[EB/OL]. http://www.ibm.com/developerworks/cn/data/library/techarticles/dm-0503aschoff/，2005.

[25] Herring C，Milosevic Z. Implementing B2B contracts using BizTalk[A]. Proceedings of the 34th Annual Hawaii International Conference on System Sciences，2002.

[26] Sybase DXP[EB/OL]. http://www.topoint.com.cn/html/fangan/2005/07/ 105732.html，2005.

[27] DT-DEP[EB/OL]. http://www.dingtian.com.cn/product/page_detail1.jsp，2007.

[28] 方正汇通数据交换平台软件[EB/OL]. http://www.echinagov.com/ echinagov/fangan/2006-4-29/3292.shtml，2006.

[29] SecExchange[EB/OL]. http://www.kind.com.cn/cpzt.jsp？id=29，2006.

[30] Infor[EB/OL]. http://publish.it168.com/2005/0922/20050922035001.shtml，2005.

[31] CenDXS[EB/OL]. http://www.ccw.com.cn/cio/solution/htm2006/20060502_116DX.asp，2006.

[32] Z Shan，X Li，Z Wang，et al. Policies and practice of e-government construction in China[A]//Proceedings of the IEEE International Conference on E-Commerce Technology for Dynamic E-Business（CEC-East'04）[C].2004.

[33] Hans J Scholl. Interoperability in e-Government：more than just smart middleware[A]//Proceedings of the 38th Hawaii International Conference on System Sciences[C]. Hawaii，2005.

[34] Chun Yu，Paul Jen-Hwa Hu. Examining the impacts of institutional framework on E-Government infrastructures：a study of Hong Kong experiences[A]//Proceedings of the 40th Hawaii International Conference on System Sciences[C]. Hawaii，2007.

[35] P J Hu，D Cui，A C Sherwood. Examining cross-agency collaborations in E-Government initiatives[A]//Proceedings of the 39th Hawaii International Conference on System Sciences[C]. Hawaii，2006.

[36] 岳昆，王晓玲，周傲英.Web 服务核心支撑技术：研究综述[J]. 软件学报，2004，15（3）：428-429.

[37] Ying Chen Lin，Sy-Yuan Li，Yuan-Shin Hwang.Dynamic load-balancing of Jini and .NET services[A]//International Conference on Parallel Processing Workshops（ICPP2006）[C]. Columbus，OH，2006.

[38] Xia Zhao，Shengxi Wu，Xingsheng Gu，et al.Research & application on information bus for MES[A]//7th World Congress on Intelligent Control and

Automation（WCICA 2008）[C]. Chongqing，2008：2654-2659.

[39] Andrew S. Tanenbaum. Computer Networks[M].Pearson Education，2003.

[40] Mark Hapner，Rich Burridge，Rahul Sharma. Java Message Service Specification v1.1 Specification[EB/OL]. http://java.sun.com/products/jms/docs.html，Sun Microsystems，12 April 2002.

[41] WebSphere MQ Java / JMS[EB/OL]. http://www.mqseries.net，2008.

[42] Oracle MessageQ[EB/OL]. http://www.oracle.com/products/middleware/tuxedo/messageq.html，2007.

[43] Andrew S. Tanenbaum. Computer Networks[M].Pearson Education，2003.

[44] Siewiorek DP，Swartz RS. Reliable System Design：the Theory and Practice. New York：Digital Press，1992.

[45] Kazunori Iwasa.Web Services Reliable Messaging TC WS-Reliability 1.1[EB/OL].http://docs.oasis-open.org/wsrm/ws-reliability/v1.1，OASIS Standard，15 November 2004.

[46] Ruslan Bilorusets，DonBox. Web Service Reliable Messaging Protocol[EB/OL]. ftp：//www6.software.ibm.com/software/developer/ library/ws-reliablemessaging200502. pdf，February 2005.

[47] Funasaka J. Evaluation on parallel downloading method using HTTP over UDP[A]//International Symposium on Autonomous Decentralized Systems（ISADS'09）[C]. Athens，2009：1-6.

[48] Hao Luo，Binxing Fang，Xiaochun Yun. Anomaly Detection in SMTP Traffic[A]//Third International Conference on Information Technology：New Generations（ITNG 2006）[C]. Las Vegas，NV，2006：408-413.

[49] HTTPR Specification[EB/OL]. http://www.ibm.com/developerworks/library/ws-httprspec/，2002.

[50] Michael R.LYU. 软件可靠性工程手册[M]. 刘喜成，钟婉懿译. 北京：电子工业出版社，1997：11-14，79-99.

[51] Coulouris G，Dollimore J，Kindberg T. Distributed Systems：Concepts and Design（Third Edition）[M]. USA MA Boston：Addison-Wesley，2001.

[52] 何国伟，王纬. 软件可靠性[M]. 北京：国防工业出版社，1998：181-183.

[53] Kanjilal，S，Chakradhar，S.T.，Agrawal，V.D. A test function architecture for interconnected finite state machines[A]//Proceedings of the Seventh International Conference on VLSI Design[C]. Calcutta，1994：113-116.

[54] 马晓轩，林学练. Web服务性能优化的研究[J].计算机工程与应用，2005，41（8）：19-22.

[55] Tim Bray，et al.，Extensible Markup language（XML）1.0[EB/OL]，World Wide Web Consortium（W3C）http://www.w3.org/TR/2004/REC-xml-20040204/，February 2004.

[56] Elfwing R，Paulsson U，Lundberg L. Performance of SOAP in Web Service environment compared to CORBA[C]. Software Engineering Conference，2002. Ninth Asia-Pacific.2002.84-93.

[57] K Devaram，D Andresen. SOAP Optimization via Parameterized Client-Side Caching[C]. In the Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems（PDCS 2003），pp. 785-790，Marina Del Rey，CA，November 3-5，2003.

[58] Seshasayee B，Schwan K，Widener P. SOAP-binQ：High-Performance SOAP with Continuous Quality Management. Distributed Computing Systems[C]. 2004. Proceedings. 24th International Conference.158-165.

[59] Noah Mendelsohn，Mark Nottingham，Hervé Ruellan. XML-binary Optimized Packaging W3C Working Draft [EB/OL]. http://www.w3.org/

TR/2004/WD-xop10-20040608/，2004.

[60] Mike Nikitas.，Improve XML Web Services' Performance by Compressing SOAP[EB/OL]. http://www.dotnetjunkies.com/Article/46630AE2-1C79- 4D5F-827E-6C2857FF1D23.dcik，2003.

[61] Erik Christensen，Francisco Curbera et al. Web Services Description Language（WSDL）1.1 W3C Note[S]. http://www.w3.org/TR/wsdl.html .W3C Group，15 March 2001.

[62] Sun Microsystems Inc.Java API for XML-Based RPC（JAX-RPC）Specification 2.0[EB/OL]. http://java.sun.com/xml/downloads/jaxrpc.html.

[63] Holt Adams .Best Practices for Web services：Part 9[EB/OL]. http://www-900.ibm.com/developerworks/cn/webservices/ws-best9/index_eng.shtml，February 2004.

[64] K Devaram，D Andresen，SOAP Optimization via Parameterized Client-Side Caching[C]. in the Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems（PDCS 2003），pp. 785-790，Marina Del Rey，CA，November 3-5，2003.

[65] Seshasayee B，Schwan K，Widener P. SOAP-binQ：High-Performance SOAP with Continuous Quality Management[C]. Distributed Computing Systems，2004. Proceedings. 24th International Conference.158-165.

[66] Noah Mendelsohn，Mark Nottingham，Hervé Ruellan. XML-binary Optimized Packaging W3C Working Draft [EB/OL]. http://www. w3.org/TR/2004/WD-xop10-20040608/，8 June 2004.

[67] F Kordon，Luqi. An Introduction to Rapid System Prototyping[J]. IEEE Transactions on Software Engineering，2002（28）：817-821.

[68] Holt Adams .Best Practices for Web services：Part 9[EB/OL]. http://www-

900.ibm.com/developerworks/cn/webservices/ws-best9/index_eng.shtml, February 2004.

[69] 马晓轩，刘旭东，林学练. 基于 Web 服务的类型映射机制的研究与实现[J]. 计算机研究与发展，2005（42）：590-594.

[70] Elfwing R，Paulsson U，Lundberg L. Performance of SOAP in Web Service environment compared to CORBA[C]. Software Engineering Conference，2002. Ninth Asia-Pacific.2002.84-93.

[71] Sun Microsystems Inc.Java API for XML-Based RPC（JAX-RPC）Specification 2.0[EB/OL]. http://java.sun.com/xml/downloads/jaxrpc.html.

[72] Microsoft Corporation. MS.Net ACT White paper[EB/OL]. http://www.microsoft.com/net/whitepapers.asp.

[73] Christian Werner，Carsten Buschmann，Tobias Jacker，Stefan Fischer. Enhanced Transport Bindings for Efficient SOAP Messaging. In Proceedings of the IEEE International Conference on Web Services（ICWS），2005.

[74] Dan Davis，Manish P arashar，Rutgers. Latency Performance of SOAP Implementations [A]//Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid[C]. 2002.

[75] Hui Liu，Xin Lin，Minglu Li. Modeling Response Time of SOAP over Http[A]. Proceedings of the IEEE International Conference on Web Services（ICWS）[C]. 2005.

[76] Kenneth Chiu，Madhusudhan Govindaraju，Randall Bramley. Investigating the Limits of SOAP Performance for Scientific Computing [A]//proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing HPDC-11[C]. 2002.

[77] 胡进锋，黎明，郑纬民，等. 带宽自适应的 P2P 网络路由协议[J]. 软件学

报，2005，16（5）：991-999.

[78] 郑纬民，胡进锋，等. 对等计算研究概论[EB/OL]. http://www.ccf.org.cn/web/resource/newspic/2005/9/20/duideng.pdf，2005.

[79] Moreno M，Matteo M，Salvatore O. Resource discovery in a dynamic grid environment[A]//Proceedings of the 16th International Workshopon Database and expert systems applications[C]. 2005.

[80] Yuhua Liu，Longquan Zhu，Jingju Gao，et al.A Segment Strategy Based on Flooding Search in Unstructured P2P Network[A]//Second International Conference on Future Generation Communication and Networking（FGCN'08）[C]. Hainan Island，2008：252-255.

[81] Zhu Y，Hu Y. Efficient，proximity-aware load balancing for DHT-based P2P systems[A]//IEEE Transactions on Parallel and Distributed Systems[C].2005：349-361.

[82] Weihong Li，Lifang Peng. Upgrade ERP from C/S to B/S based on Web service[A]. 2005 International Conference on Services Systems and Services Management（ICSSSM'05）[C].2005（1）：593-597.

[83] Vishal Iyengar，Sameer Tilak，Michael J.Lewis，Nael B.Abu-Chazaleh. Non-uniform information dissemination for dynamic grid resource discovery [A]//Proceeding of the Third IEEE International symposium on network computing and applications[C]. 2004.

[84] S Tilak，A Murphy，W Heinzelman. Non-uniform information dissemination for sensor networks[A]//Proceeding of the 11th IEEE International Conference on Network Protocols（ICNP'03）[C]. 2003.

[85] Z Du，J Huai，Y Liu，et al. IPR：an Automated Business Process Reconciliation[A]//Proceedings of International Conference of IEEE/WIC/

ACM Web Intelligence（WI05）[C]. 2005.

[86] Daniel Austin，Abbie Barbir，Sharad Garg. W3C Web Services Architecture Requirements[EB/OL]. http://www.w3.org/TR/2002/WD-wsa-reqs-20020429，2002.

[87] Heather Kreger. Web Services Conceptual Architecture（WSCA 1.0）[EB/OL]. IBM，2001.

[88] UDDI Version 2.04 API Specification UDDI Published Specification[EB/OL]. http://uddi.org /pubs/ ProgrammersAPI- V2.04-Published-20020719 .pdf，2002.

[89] 马晓轩，怀进鹏，王芝虎.基于 UDDI 的应用服务注册中心的设计与实现[J]. 北京航空航天大学学报，2005，31（9）：1040-1044.

[90] Java TM API for XML Messaging（JAXM）[EB/OL]. http://java.sun.com/xml/jaxm/，2003.

[91] 葛声，胡春明，杜宗霞，等. 基于 Web Service 的应用支撑环境研究与实现[A]//2002 全国软件与应用学术会议（NASAC）论文集[C]. 北京：机械工业出版社，2002：97-102.

[92] UDDI4J[EB/OL]. http://uddi4j.sourceforge.net/，2006.

[93] Erich Gamma，等. 设计模式——可复用面向对象软件的基础[M]. 北京：机械工业出版社，2007.

[94] CapeClear 4 Technical Overview[EB/OL]. http://www.capeclear.com/products/whitepapers/index.shtml，December 2002.

[95] Jun Han，Xudong Liu，Jinpeng Huai，Xuelian Lin，Xiaoxuan Ma，Xian Li：InfoXP：An E-government Information eXchange Platform Based on Overlay Network[A]//ACIS-ICIS[C]. Australia，2007：949-954.

[96] Raman，L. OSI systems and network management[J]. Communications

Magazine，IEEE，1998，36（3）：46-53.

[97] ebXML[EB/OL]. http://www.ebxml.org，2006.

[98] Gnutella[EB/OL]. http://www.gnutella.com，2007.

[99] Rowstron，P. Druschel. Pastry：Scalable，distributed object location and routing for large-scale peer-to-peer systems[A]//Proc. of 18th IFIP/ACM International Conference on Distributed Systems Platforms（Middleware'01）[C]. Heidelberg，Germany，November 2001.

[100] Stoica，R.Morris，D.Karger，et al. Chord：A scalable peer-to-peer lookup service for Internet applications[A]//Proceedings of ACM SIGCOMM' 2001[C]. USA，August 2001.

[101] Clarke，O. Sandberg，B. Wiley，et al. Freenet：A Distributed Anonymous Information Storage and Retrieval System[A]//Proc. of the ICSI Workshop on Design Issues in Anonymity and Unobservability[C]. Berkeley，CA，2000.

[102] Al-Shaer，E. Managing firewall and network-edge security policies[A]// Network Operations and Management Symposium（NOMS）[C]. South Korea，Vol.1，2004：926.

[103] Wenjie Wang，Cheng Jin，Sugih Jamin. Network Overlay Construction under Limited End-to-End Reachability[A]//Proceedings of IEEE INFOCOM 2005[C]. USA，2005.

ISBN 978-7-5111-2839-3

中国环境出版社
天猫旗舰店

定价：26.00元